ATL-QOS: AN ADAPTIVE TRUST-AWARE LOCATION-BASED FRAMEWORK FOR ACHIEVING QOS IN MANETS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2016

By
Helen Bakhsh
School of Computer Science

Contents

Word Count: 51295

List of Tables

List of Figures

List of Algorithms

Abstract

Mobile ad hoc networks (MANETs) have gained increasing attention from industry for their potential applications. MANETs allow devices to communicate in areas with no pre-existing communication infrastructure. In MANETs, node mobility leads to dynamic changes in network topologies and fluctuations in network available bandwidth. The lack of infrastructural support means that communication nodes need to collaborate among themselves functioning as routers (intermediate nodes) for other nodes. This places additional processing and communication loads onto the communication nodes and opens up doors to more active attacks by intermediate nodes. It is also worth noting that mobile nodes are typically battery powered, and they are more restrictive in terms of storage space and processing capabilities than their wired counterpart. These MANET features indicate that achieving *QoS* in MANETs should be done in the most cost-effective manner.

In this thesis, a novel Adaptive Trust-aware Location-based (*ATL-QoS*) framework is proposed. The *ATL-QoS* framework can harvest fluctuating available bandwidth in the underlying network to deliver high priority traffic in various network conditions. The novelty of the framework lies in that it uses single path and multiple path deliveries and packet duplication over multiple path, in an adaptive manner, in an attempt to increase high priority traffic delivery with minimum bandwidth overhead costs. The framework handles low and high priority traffic in a differential manner. To implement these ideas, two novel *ATL-QoS* components are designed: (1) a Trust-Aware Dynamic Location-based (improved version) ($TADLV2$) multiple path discovery protocol and (2) a path Selection, traffic Allocation, and path Verification (SAV) solution. The $TADLV2$ protocol is designed to discover multiple path between a pair of communication nodes with minimum bandwidth overheads, we first designed $TADL$ protocol and then an improved version of $TADL$, $TADLV2$. The SAV solution is designed to increase high priority traffic delivery success.

These *ATL-QoS* framework ideas are implemented and evaluated using the *NS-2* simulation and compared against the most relevant protocol in the literature. The simulation study shows that *ATL-QoS* outperforms the relevant protocol in terms of reducing routing overheads and increasing packet delivery ratios. These enhancements making *ATL-QoS* more effective in providing *QoS*.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

Thanks to Allah, who gave me the chance for learning; gave me the support and love.

I would like to express my special appreciation and thanks to my supervisor Dr. Ning Zhang, for her constant help and guidance. She has been helping me out and offered me so much advice, patiently supervising me, and always guiding me in the right direction. I have learned a lot from her, without her help I could not have finished my thesis successfully.

I would also like to thank The University of King Abdulaziz for giving me this opportunity and providing me with a full scholarship to complete my PhD studies. In addition, thanks to the Saudi Ministry of Higher Education for their services and support.

A special thanks to my family. Words cannot express how grateful I am to my mother, and father for supporting me spiritually throughout writing this thesis. To my lovely kids, I would like to express my thanks for always cheering me up. To my beloved husband for his patience, support and sacrificing his time to come here to Manchester so that I can complete my study.

Last but not least, thanks to The University of Manchester for providing excellent learning environment and assist the students to achieve their goals.

Abbreviations

ABE	Available Bandwidth Estimation
ABW	Available BandWidth
ACK	ACKnowledgement
ADQR	ADapive QoS Routing
AODV	Ad hoc On Demand Distance Vector
AODV-BR	AODV Backup Routing
AODVM	Ad hoc On-demand Distance Vector Multi-path
AOMDV	Ad hoc On-demand Multi-path Distance Vector
A2SA	Adjusting Starting Searching Area
ATL-QoS	Adaptive Trust-aware Location-based QoS
BPS	Best Path Selection
CAMRLB	Congestion Adaptive Multi-path Routing protocol for Load Balancing
CBQR	Cluster-Based QoS Routing
CBR	Constant Bit Rate
CPU	Central Processing Unit
CRESQ	Cluster-based Routing for End-to-end Security and QoS
CTR	ConTRol
2-DAARC	2-Dimensional Adaptation ARChitecture
DFB	Destination FeedBack
DiffServ	Differentiated Services
DPC	DFB Packet Creation
DPL	Destination node Predicted Location
DREAM	Distance Routing Effect Algorithm for Mobility
DRS	Data packets Received and Sent

DSR	Dynamic Source Routing
ETX	EXpected Transmission
FQMM	Flexible QoS Model for MANETs
GPS	Global Position System
GPSR	Greedy Perimeter Stateless Routing
IC-MNPD	Intermediate node Controlled Multiple Node-disjoint Path Discovery
IETF	Internet Engineering Task Force
IntServ	Integrated Service
ISQRDC	Integrated Security and QoS Routing and Data Communication
KAU	King Abdulaziz University
LAR	Location-Aided Routing
MA-A2SA	Mobility-based Adaptive Adjusting Starting Searching Area
MA-MNPD	Mobility-based Adaptive Multiple Node-disjoint Path Discovery
MANET	Mobile ad hoc network
MPS	Multiple Path delivery Service
MRP	Multi-path Routing Protocol
MSR	Multi-path Source Routing
NNI	Neighbouring Node Information
NS-2	Network Simulator version 2
OLSR	Optimized Link State Routing
OQR	On-demand QoS Routing
PDR	Packet Delivery Ratio
PLBQR	Predictive Location-Based QoS Routing
PQA	Path Quality Adjusting
PSS	Previous Searching area Size

QLB-AOMDV	QoS and Load Balancing AOMDV
QOLSR	QoS extension of the OLSR
QoS-AODV	Quality of Service AODV
QoS-GPSR	QoS Greedy Perimeter Stateless Routing
QPS	QoS Paths Selection
QTA	QoS Traffic Allocation
RERR	Route ERRor
RMSR	Redundant Multi-path Source Routing
RREP	Route REPly
RREQ	Route REQuest
RSN	RREQ Sequence Number
RSR	Redundant Source Routing
SAV	path Selection, traffic Allocation and path Verification
SAS	Searching Area Size
SC-MNPD	Source node Controlled Multiple Node-disjoint Path Discovery
SMR	Split Multi-path Routing
SMT	Secure Message Transmission
SPS	Single Path delivery Service
SQ-AODV	Stability-based QoS-capable AODV
TADL	Trust-Aware Dynamic Location-based
TADLV2	Trust-Aware Dynamic Location-based Version 2
TBRPF	Topology Broadcast Reverse Path Forwarding
TDR	Trigger-based Distributed QoS Routing
TNS	Trust-based Neighbouring Selection
TOHIP	TOpology-HIding multi-path Protocol
Trusted-DSR	Trusted Dynamic Source Routing
TRV	TRust Value
TVE	Trust Value Estimation

Chapter 1

Introduction

1.1 Background and Motivation

A MANET [121] is a self-configuring wireless network that is formed dynamically by a collection of wireless nodes. The nodes are typically mobile, resulting in frequent changes in network topologies. As a result, a path established a moment ago may not be valid now. If a transmission node (i.e., a source node) is not able to respond to such dynamic topological changes in a timely manner, it will not deliver the traffic properly, e.g., the traffic may experience excessive delays or even packet loss. The frequent changes in the network topology lead to fluctuations in the bandwidth available in the network.

A MANET does not have any infrastructural support, so to provide the network functionality, it relies on the collaborations among communication nodes themselves. In other words, the nodes in a MANET should collaborate to deliver their traffic to the destinations. This not only places additional processing and communication loads onto the communication (intermediate) nodes but also opens the door to more active attacks by intermediate nodes [69]. The additional loads and potential active attacks by the intermediate nodes can have detrimental effects on Quality of Services (QoS).

In addition, the open wireless media make the network vulnerable to passive attacks. To protect against these attacks, security mechanisms are usually used, and these mechanisms impose additional computational and communicational overheads, which, in turn, would deplete QoS.

It is also worth noting that mobile nodes are typically battery powered, and they are more restrictive in terms of storage space and processing capabilities than their wired counterparts [118]. In other words, a wireless mobile node usually has a lower level of resource capabilities (CPU, memory, etc.) than a desktop or laptop computer. Hardware failures and depleted batteries, for example, can lead to path unavailability and routing disruption. This affects the *QoS* a network could provide to the traffic.

MANETs have gained increasing attention for their potential applications. A MANET can either operate as a stand-alone network or as part of an integrated network [71, 38]. A stand-alone MANET allows devices to communicate in areas with no pre-existing communication infrastructure. A typical example of a stand-alone MANET is one established to support emergency search-rescue operations in disaster relief. Rescue workers can send rescue robots, controlled over a MANET remotely, into a damaged building. A robot can send streaming video and static images to rescue workers so that they can assess the disaster scene. Another example is on the battlefield to support military communications between moving vehicles and/or soldiers. A soldier may communicate with his commander using a real-time video feed over a MANET.

A MANET may be integrated with another infrastructure-less network, or may be used as an accessing network for the Internet. This integration brings benefits for both networks [25]. It allows the MANET nodes to reach out to wider application domains and services and the infrastructure network to be extended to cover a wide area. In other words, MANETs can provide opportunistic connections in areas that are not covered with infrastructure networks and/or if users do not have access to those networks. This interconnection can be achieved by using gateways, which act as bridges between a MANET and the Internet. A typical example of the use of an integrated MANET is in medical health care. Patients can access medical expertise or expert treatment anywhere and at any time. Expert treatment or diagnoses are possible while a patient is on his way to a hospital in an ambulance, or in places where there is a lack of expertise. In such a scenario, real-time video, X-rays, and electronic patient records can be transmitted, so that experts, regardless of their locations, can make a visual inspection of these data and provide a diagnosis to patients who need them.

From the application scenarios described above, two observations can be made. The first observation is that there are different types of data transported in the network, e.g., protocol control data, device control data, and user data, and for user data, there may be text data, video data, image data, or voice data.

Different data types may have different priorities or may impose different *QoS* requirements. For example, the protocol control data should have the highest priority because they are important for maintaining network operations. Real-time video from robots used in a disaster relief operation may require priority treatment from the network. *QoS* [99] is the ability to provide differential services to different applications and/or different streams in an application. The purpose of providing *QoS* is to satisfy users'/applications' *QoS* requirements with minimum overhead costs in term of processing time, memory space, battery, and bandwidth consumption. To support the *QoS* requirements of the different data types effectively, it is necessary to consider MANET characteristics. The two main notable characteristics are the frequent link breaks and MANETs limited resources. The frequent link breaks cause fluctuating available bandwidth and packet loss in the network. The limited resources in MANET indicated that bandwidth, battery life, storage space, and processing capability in the network are not as ample as in a wired network, therefore, any *QoS* solution designed should be cost efficient, introducing as low as possible computational complexity and bandwidth overheads.

The second observation is that as MANETs use wireless media and there is a lack of any infrastructure support, the network is vulnerable to a range of security attacks [69, 81, 32]. In this research, we only consider packet dropping attacks. This type of attack includes black and grey hole attacks. In black hole attacks [93, 116, 81], attacker drops packets (data packets or control packets) to preserve their resource or to deliberately disrupt network operations. In grey hole attacks [93], attacker drops packets selectively. *QoS* can be affected by these attacks. So it is necessary to consider these attacks when designing *QoS* solutions for MANETs.

1.2 Problem Statement

This project is set to investigate how to best support *QoS* in the presence of node mobility and packet dropping attacks in a MANET environment. This requires investigation of how to discover multiple path in the most efficient manner, how to make the best use of the fluctuating bandwidth offered by the discovered paths and how to reduce detrimental effects of packet dropping attacks on QoS.

In working on this problem, we assume that source and destination nodes are mutually trustworthy, and the intermediate nodes are not trustworthy. Trust here means the reliability of a link or a path in delivering packets. Reliability measures in terms of the ratio of data packets that are successfully delivered to their intended destination. Each node estimates a trust value for each of its neighbouring nodes based on the reliability of the link connecting to the neighbouring node.

1.3 Research Aim and Objectives

The aim of this research is to find a cost-efficient and network layer solution to *QoS* provisioning in MANET to support the delivery of high priority traffic in the presence of node mobility and packet dropping attacks in this environment.

To achieve this aim, the objectives of the research are as follows:

- Investigate MANET characteristics and security threats in this environment, and identify challenging issues in achieving *QoS* in the MANET environment.
- Identify efficient ways to address the challenging issues and this includes the following:
  - Investigate and analyse existing MANET *QoS* solutions and understand weaknesses and missing features in the solutions.
  - Investigate mechanisms used for protecting against packet dropping attacks.
  - Specify design requirements for a *QoS* solution to support effective and efficient *QoS* provisioning in a dynamic MANET containing packet dropping attackers.
- Design an efficient *QoS* solution satisfying the requirements specified above.
- Evaluate the designed solution to study its effectiveness and efficacy, and to compare evaluation results with the most relevant related work.
- Publish research results to solicit feedbacks from experts in the area.

1.4 Challenging Issues

Providing QoS for high priority traffic in a MANET containing packet dropping attackers with as low cost as possible is a challenging task. This is because, firstly, the dynamic changes in the network topologies, caused by node mobility, will lead to fluctuating bandwidths in the network. To tackle this challenging issue, we need to find answers to the following questions: (a) how to discover the most reliable path, i.e. how to select the most reliable set of nodes to form a path; (b) how to discover node-disjoint paths; (c) how to achieve (a) and (b) with the lowest cost possible, (i.e. lowest possible communication overhead introduced into the network and as less computational complexity imposed on the network nodes as possible); (d) how to make the best use of the fluctuating bandwidth.

The second challenging issue is how to reduce trust imposed on the intermediate nodes. As MANETs use open wireless media and rely on collaboration among communication nodes, they are particularly vulnerable to security attacks. Some of the attacks have direct negative effects on QoS. For example, some intermediate nodes may be compromised such that they became malicious (drop data packets) or non-cooperative (selfish nodes). The question is how to provide QoS without requiring trust of the intermediate nodes, or if this is impossible, how to reduce the trust imposed on the intermediate nodes.

1.5 Novel Contributions and Publications

The work presented in this thesis has six novel contributions.

Novel Contribution 1: Three ideas used to reduce bandwidth cost in discovering multiple path

Bandwidths are essential for provisioning of *QoS*. Reducing bandwidth costs in path discovery can have a positive effect on *QoS* provisioning. Three ideas have been identified and applied to our path discovery process. They are: (1) using location-based approach to path discoveries; (2) dynamically resizing the broadcasting area; and (3) using a local trust model. To find multiple path, nodes need to send route discovery request packets, and these packet will be broadcast to its neighbours. Every neighbour receiving a packet will further broadcast it to their neighbours. We need a way to reduce the number of route request packets board into the network; we confine the forwarding of request packets within a certain area containing the source and destination nodes. This is called the directional approach. The size of this area, the search area, determines the number of route request packets poured into the network as well as the number of paths that can be discovered. The bigger the search area, the more paths may be found. This also means that a higher bandwidth cost will be introduced. To reduce the bandwidth cost, we dynamically adjust the size of this search area for a sufficient number of paths required by the traffic flow. An ad hoc network is a spontaneous network, and not all the nodes are trustworthy; if a malicious node is included in the path it may drop packets. In other words, some nodes in the network may be more trustworthy than others, and the more trustworthy the nodes that are involved in the path, the more reliable the traffic delivered. This means that we need to find a way to evaluate the trustiness of nodes and to select nodes based on their trust values. This leads us to embed a trust model to govern the selection of the intermediate nodes in path formation. To reduce the cost incurred with trust evaluation into our solution, we have used a local trust model.

Novel Contribution 2: Novel algorithm for dynamic resizing of search area size

As mentioned earlier, the number of paths that can be discovered depends on the size of the search area used. The smaller the search area size, the fewer paths may be discovered, and the less overhead may be introduced into the network. So it is intuitive to use a smaller search area size when we need fewer paths. Similarly, we should enlarge the search area size when we need a larger number of paths. However, the number of discovered paths also depends on other factors. We need a method to resize the search area taking into account all the relevant factors and to do so without introducing bandwidth cost. This leads to a novel algorithm for dynamic resizing of the search area size. Novel contributions 1 and 2 have been built into the Trust-Aware Dynamic Location-based $TADL$ protocol [10].

Novel Contribution 3: A more efficient algorithm for node-disjoint multiple path discovery

Through literature research, we discovered that node-disjoint paths provide the best packet delivery reliability. In TADL, we have implemented a node-disjoint path discovery method by which the node first discovers all paths linking the source and destination nodes within the specified search area and then selects the most trustworthy set of node-disjoint paths from the discovered paths to deliver the traffic. However, through a simulation study of $TADL$, we have discovered that this approach introduces a higher level of overhead in the network in case of a high mobility network. We, thereafter, revised the method, and this has led to our novel contribution, i.e. a more efficient multiple node-disjoint paths discovery method, and this method has implemented in a $TADLV2$ protocol [11] (An Improved Version of $TADL$).

Novel Contribution 4: An adaptive algorithm to harvest the fluctuating available bandwidths from multiple path to support high priority traffic delivery

In the $TADLV2$ protocol design, we use paths' trust values to govern the selection of the most trustworthy path from the discovered paths to route the traffic. However, due to node mobility, the available bandwidth provided by a path is fluctuating, and it is not always possible to find a single path that is both trustworthy and has the required bandwidth to support the delivery of a high priority traffic flow. Therefore, there is a case for making use of the available bandwidths provided by multiple path. This leads to the use of the available bandwidths provided by multiple path, and duplicate packets transmitted over low trustworthy paths to improve traffic delivery reliability in an adaptive manner.

Novel Contribution 5: An adaptive algorithm for adjusting traffic allocation decision to support high priority traffic delivery

During a delivery phase, the available bandwidth and the trust values of the paths used may change. This is because when nodes move, their available bandwidth will change. The untrustworthy intermediate nodes may drop packets causing trust value changes and nodes may lie about their available bandwidth values collected during path formation phase. Therefore, we should find a way to verify the bandwidth and trust values of a used path during the delivery phase and to reduce or increase these values, if necessary. This leads to the use of the destination node feedback mechanisms, which allows the destination node to report back to the source node the number of data packets received by the destination node. The source node then uses this feedback to validate and adjust the path's available bandwidth and trust values discovered during the path formation phase. In this way, the intermediate node could only lie about the available bandwidth for a short period of time, i.e. between when the first packet is transmitted to when the destination feedback is received. Novel contributions 4 and 5 have been implemented into the path Selection traffic Allocation and path Verification *SAV* solution.

Novel Contribution 6: ATL-QoS Architecture Design

We have also designed the *ATL-QoS* framework that captures all the functions required for the delivery of traffic and the provision of *QoS* services to the traffic delivery. These include what we have described above, i.e. multiple node-disjoint path discoveries, traffic duplications, traffic allocations on multiple path, and the delivery and reception of traffic with two different priorities. This design is presented in detail in chapter 3.

Parts of the research work presented in this thesis have been published in the following conference proceedings.

- Helen Bakhsh, Ning Zhang, and Andy Carpenter. TADL: A Trust-Aware Dynamic Location-based protocol suite for discovering multiple path in MANETs. In Proceedings of the 2015 International Conference on Distributed Computing and Networking, ICDCN '15, pages 2: 1-2: 10, New York, NY, USA, 2015. ACM.
- Helen Bakhsh, Ning Zhang, and Andy Carpenter. TADL-V2: An improved Trust-Aware Dynamic Location-based adaptation protocol for discovering multiple path in MANETs. EAI Endorsed Transactions on Ambient Systems, 15(6), 8 2015.

1.6 Thesis Structure

The remainder of this thesis is organised as follows.

Chapter 2 provides a survey and critical analysis of the related work in MANET, which is relevant to the *QoS* context. Based on the identified strengths and weaknesses of the existing solutions, the chapter outlines the missing gaps in these solutions. This chapter also includes a discussion of the challenging issues in designing a routing protocol for MANETs. Finally, this chapter makes recommendations for the best way forward. These recommendations are used in the design of the novel solution in Chapter 3.

Chapter 3 presents the design of our novel solution, the *ATL-QoS* framework. It first specifies the requirements needed for the design of an efficient *QoS* solution for MANETs containing packet dropping attackers, along with measures taken to satisfy these requirements. It then describes the architecture of the framework and the architectural components. It also gives a run-through of the operation of the framework.

Chapter 4 presents the building blocks used in the design of the *ATL-QoS* framework. The building blocks include the Trust Value Estimation (TVE) method [88], and the Available Bandwidth Estimation (ABE) method [22]. The chapter also outlines the evaluation methodology used and justifies the use of a simulation as the evaluation methodology. It also describes the simulation configuration and presents the validation of the *ATL-QoS* framework using simulation studies.

Chapter 5 presents the design, implementation, and performance evaluation of the $TADL$ protocol. The $TADL$ protocol is a multiple path discovery protocol. It uses a location-based adaptive directional paths discovery approach to reduce the risk of flooding control packets into the underlying network. The $TADL$ protocol is evaluated against the LAR routing protocol [52]. Shortcomings of the $TADL$ protocol are identified.

Chapter 6 presents the design, implementation, and performance evaluation of the $TADLV2$ protocol, which is an improved version of the $TADL$ protocol. The novelty of the $TADLV2$ protocol that it can accommodate dynamic topological changes of the network in an adaptive manner. The $TADLV2$ protocol is evaluated against the $TADL$ protocol.

Chapter 7 presents the design, implementation, and performance evaluation of the SAV solution. The SAV solution is a multiple path selection and traffic allocation protocol aimed at delivering high priority traffic in a MANET environment in a more reliable manner. The protocol can harvest fluctuating available bandwidth in the underlying network to deliver high priority traffic under various network conditions. It is also used to validate and adjust the path's quality discovered during a path formation phase. The solution is aimed at supporting dynamic adjustment of high priority traffic deliveries in response to the fluctuations in the available bandwidth of the path used and any dishonesty by the intermediate nodes included in the paths. The SAV solution is evaluated against the $TADLV2$ protocol.

Chapter 8 summarises the thesis and highlights major findings in the thesis. This includes the issues found during the research. It also includes the discussions for future work.

Chapter 2

Achieving QoS in MANET: Literature Survey

2.1 Chapter Introduction

This chapter gives a survey of the MANET routing protocols and the-state-of-the-art *QoS* solutions at the network layer. The solutions can be classified into Non-MANET and MANET solutions. The non-MANET solutions can further be classified into two groups: resource reservation and service differentiation based approaches. The MANET solutions can also be classified into two groups: *QoS* frameworks and *QoS* routing protocols. This chapter critically analyses the state-of-the-art solutions to identify their strengths and limitations, so that in the design of our proposed solution, we can maintain the strengths while overcoming their weaknesses. This chapter also discusses the impacts of security threats on *QoS* routing protocols and the existing schemes proposed to tackle them. Through this survey, the chapter also identifies areas of improvement for *QoS* provisioning in a MANET environment.

Section 2.2 surveys the existing approaches to MANET routing in detail. The resource reservation-based approaches are discussed in Section 2.3. Section 2.4 surveys the existing approaches to service differentiation. Section 2.5 discusses *QoS* frameworks. Other existing *QoS* routing protocols are discussed in Section 2.6. Section 2.7 describes the security threats to *QoS*. The existing ways to counter these threats are analysed in Section 2.8. Section 2.9 discusses existing *QoS* methods that consider *QoS* and security issues in an integrated manner. Section 2.10 presents our vision in terms of providing *QoS* in MANETs and the challenging issues in this context. Section 2.11 outlines a way forward to achieve *QoS* in MANET. Finally, Section 2.12 concludes the chapter.

2.2 MANET Routing Protocols

MANET routing protocols proposed in the literature can largely be classified into two categories, topology-based and location-based routing protocols. They are discussed in detail in the following subsections.

2.2.1 Topology-Based Protocols

Topology-based routing protocols [74] seek paths based on network topological information. There are many such protocols published in literature, but the major ones have been recognised by the Internet Engineering Task Force (*IETF*) MANET Working Group [16]. These protocols can largely be classified into two groups: proactive and reactive ones.

A proactive routing protocol [2, 98] discovers and updates routing information periodically regardless whether the routes will be used or not. Every node in a network maintains routing information to every other node in the network. This information is discovered and updated periodically through the use of control packets. The Optimized Link State Routing (*OLSR*) [41, 43, 79] and the Topology broadcast reverse path forwarding (*TBRPF*) [83, 2] protocols are two examples of proactive routing protocols.

This class of routing protocols has an advantage, i.e. when a node is to transmit or forward data packets, routing information is already available, thus, the data packets can be transmitted with little delivery delays and a high delivery ratio may be achieved. However, this approach has three limitations. Firstly, it introduces a high level of routing overhead due to the periodic exchange of routing updates among nodes even in the absence of data traffic or topological changes. The level of routing overhead is proportional to the network size; it is independent of the traffic load in the underlying network.

Secondly, this approach imposes a high level of storage requirement on each node in the network as each node needs to maintain routing information for every other node in the network in its routing table. In addition, when the nodes' mobility level is high, the underlying network topology will change more frequently, so the routes maintained in the routing tables may soon become invalid. To address this problem, the time interval between route updates should be reduced. This will further increase the control traffic injected into the underlying network, which will make the network more prone to congestion. As a result, the packet delivery ratio will decrease, and the data packet delivery delays will increase.

With the reactive routing protocol [111, 2, 98], on the other hand, a source node only discovers routes when it has data packets to send. When discovering a route, the source node floods the entire network with Route Request ($RREQ$) packets until the destination node, or an intermediate node with the latest information about the destination, receives the $RREQ$ packet. The destination node, then, generates and returns a route reply ($RREP$) packet, that contains the routing information for forwarding the data packets, to the source node. In other words, with this approach, a route discovery process is performed in an on-demand manner. The Ad hoc On Demand Distance Vector ($AODV$) [111, 20, 97] and the Dynamic Source Routing (DSR) protocols [111, 117, 46] are examples of this approach.

The advantage of this class of protocols is that a path discovery process is only invoked in response to communication needs, and this approach introduces lower level of routing and storage overheads than the proactive approach. However, with the reactive approach, the data packet delivery delays are usually higher than the proactive approach. This is because, with the reactive approach a source node may need to discover a route after it has data packets to send.

In general, topology-based routing protocols scale in a small network (say consisted of a few hundred nodes) [15]. When the network size goes larger, they exhibit poor performance. Also, topology-based routing protocols are not cost effective for a highly mobile network as, to discover routes, these protocols broadcast control packets to all the nodes in the network. When the nodes' mobility level is high, an excessive quantity of control packets are poured into the network causing a high level of routing overhead in discovering new routes or maintaining up-to-date network topology information. In addition, both proactive and reactive routing protocols only offer a best effort routing service. They do not provide any QoS, i.e. they don't provide any traffic priority differentiation.

2.2.2 Location-Based Protocols

Location-based routing protocols [52] make use of the network nodes' location information to restrict the flooding of route discovery and/or data packets to the whole network while discovering routes. The routing decision is based on the destination node's location and the location of its neighbouring nodes. This class of protocols requires that each node in the network be able to determine its own physical location and the locations of a destination node. This location information is used to guide the control packets and in some cases data packets towards the direction of the destination node.

There are two aspects where the location-based approach differs from the topology-based approach [74]. These are the location service and the packet forwarding process. The location service is used by a source node or an intermediate node to determine the physical location of network nodes or/and a destination node before routing or transmitting a control or data packet. Each node maintains a location table, containing the geographical positions of all or some of the nodes in the network. Each node determines its own physical position through the use of a location service, such as Global Position System (*GPS*). It then uses one of the two ways to propagate the location information to other nodes in the network. The first way is to broadcast the location information periodically to all the other nodes in the network. The advantage of this approach is that each node will have the updated location information about all the other nodes in the network when it has data to transmit. However, there are two drawbacks with this approach. Firstly, the size of the location table is proportional to the number of nodes in the network. Secondly, broadcasting the location information periodically can increase the routing overhead in the network.

The second way is to propagate the location information during route discovery and data transmission phases. In this case, the location information of each intermediate node along a path will be added into a passing packet. Upon receiving the packet, an intermediate node in the path will learn the location information of all the other nodes in the path. The advantage of this approach is that no additional packets are used to convey the location information, thus reducing the overhead in the network. However, the location information maintained in the nodes' location tables may not be up-to-date.

The second difference in this location-based approach is that, when a node transmits or forwards a control or data packet, it only transmits or forwards it towards the direction of the destination node. This is different from the approach taken by the topology-based protocols, where packets are broadcast across the entire network. For packet forwarding in the location-based approach, one of the two main packet forwarding strategies [74, 92] may be used: a restricted directional forwarding strategy and a greedy forwarding strategy. With the restricted directional forwarding strategy when searching for a path to a destination node, a node floods the control packets to the nodes in a search area towards the direction of the destination node. The discovered path is then used to transmit the data packets. With the greedy forwarding strategy, on the other hand, a node immediately forwards data packets to one neighbour that is located closer to the destination node than the forwarding node itself, without using control packets to discover paths. Also, with the greedy forwarding strategy, the selection of the downstream neighbour is done by using the optimisation criteria, such as the next downstream neighbour should be the closest neighbour in distance to the destination node. Location-Aided Routing (LAR) [52, 74], Distance Routing Effect Algorithm for Mobility ($DREAM$) [13], and Greedy Perimeter Stateless Routing ($GPSR$) [48] are examples of the location-based approach.

LAR is a location-based reactive routing protocol which uses the restricted directional forwarding strategy. To find paths, a source node floods a control packet to all the nodes that are located in a search area. The search area is defined by using the location information of a destination node. An intermediate node, upon the receipt of the control packet, floods the packet to all the nodes that are located in the search area until the destination node is reached. The source node then uses the first discovered path to transmit data packets.

$DREAM$ is a location-based proactive routing protocol that uses the greedy forwarding strategy. $DREAM$ requires that each node in the network to maintain the location information of each other node in the network. Similar to LAR, a source node sends packets to all its neighbouring nodes in the direction of the destination node. However, in $DREAM$, any sent packets are data packets not control packets. Each of these neighbours then defines their own search area of the destination node, based on their own copy of the destination node's location information, and forwards the packet accordingly until the destination is reached.

The $GPSR$ is a location-based reactive routing protocol that uses the greedy forwarding strategy, where the next hop is the closest node, in distance, to the destination. Similar to $DREAM$, it does not require a source or an intermediate node to search for a path prior to data transmission. It simply uses the neighbouring nodes' location information to decide where to forward the data packets. However, different from *DREAM*, where each intermediate node uses its own copy of the destination node location information to make a routing decision, in *GPSR*, a source node puts the physical location of the destination node into the outgoing data packet. When an intermediate node receives the packet, it forwards it to the neighbour that is closest to the destination node. This process is repeated at each intermediate node until the destination is reached.

The location-based routing protocols have two main advantages. The first is that the communication/routing overheads they introduce are lower than topology-based protocols. This makes them scale better in a large network than topology-based routing protocols. The second one is that this approach has a higher data packet delivery ratio due to the reduced routing overhead.

However, both classes of protocol, topology-based routing and location-based routing protocols, process packets on an equal basis. They do not differentiate priority traffic from non-priority ones. They provide a best effort service to the delivery of all the packets. The best effort service does not provide any form of special support to high priority traffic, thus, cannot satisfy any special *QoS* requirements the high priority traffic may have [99, 35, 77].

2.3 Resource Reservation

A reservation-based approach [59, 123, 95, 16] is an approach intended for providing a hard *QoS* guarantee for a high priority traffic flow. It achieves this by reserving resources (typically bandwidths) required by the traffic flow on the nodes involved in a path from the source to the destination nodes prior to the delivery of the traffic flow. The resource reservation means that each node along the path must reserve resources for each flow and isolate each flow from the others. This requires each node maintain the state information for each of the flows reserved by the node. The Integrated Service (*IntServ*) model [7, 108, 131] is based on the reservation-based approach. It provides an end-to-end *QoS* guarantee on a per-flow basis, i.e. the resources reserved for a flow are guaranteed for the duration of the session established for the flow.

This reservation-based approach is not suited to MANET environments [29, 7, 108]. This is because it uses separate control packets to reserve the resources and these control packets not only consume bandwidth but also are targets of security attacks. Also, the level of information maintained at an intermediate node is proportional to the number of flows maintained by the node. For a large network, this approach is not scalable. Furthermore, in a highly mobile network where the network topology changes dynamically, a reservation may become void before the data delivery is completed. When this happens, a new reservation along a new path will be necessary. These repeated path reservations will lead to an increased number of reservation packets being injected into the underlying network, leading to an increased level of bandwidth consumption and a lower level of *QoS*. Also, as mentioned above, reservation packets themselves could be the targets of security attacks. Protecting them will impose a higher processing overhead on each node. All these factors will have a negative effect on *QoS*.

2.4 Service Differentiation

Service differentiation supports packet delivery for different priority levels. This approach provides *QoS* on a per-class basis rather than per-flow basis as in the case of the resource reservation-based approach. Service differentiation was designed to overcome the scalability problem seen in the resource reservation-based approach. Instead of using resource reservations, here, *QoS* is provided through the aggregation of traffic flows. It aggregates flows with similar traffic behaviours into classes, and forwards each class of packets based on their *QoS* requirements. The Differentiated Services ($DiffServ$) model [7, 108, 131, 16] implements this differentiation approach. It divides the nodes in a network into edge routers and core routers. It assigns all the complex processing tasks to edge routers and makes the core routers focus on forwarding data packets only. The edge routers are responsible for the classification of flows and mark the packets so that they receive differentiate treatments at the core routers. $DiffServ$ supports two kinds of services [94]: a premium service and an assured service. The premium service is a higher priority service than that offered by the assured service. The premium service is a guaranteed service proposed for traffic that requires *QoS* guarantees. It is aimed at providing a better than "best-effort" service to serve high priority traffic at times of congestion. The better than best-effort service is provided under light to moderate network loads, where the assured class provide the required *QoS* for every flow whenever the resources are available. When congestion occurs, the assured class can be demoted to a lower priority class, or it can be dropped, with the additional bandwidth being used to guarantee the *QoS* of high priority (or premium) traffic.

The Service differentiation has two advantages [16]. Firstly, no admission control or signalling packets are required at each core router, so the routing overhead and processing complexity imposed on the core router are lower than the resource reservation-based approach. Secondly, the state information managed inside the network is limited by the number of classes instead of the number of flows or streams (as in the case of *IntServ*). This makes the *DiffServ* more scalable than the *IntServ*.

However, the service differentiation approach also has drawbacks [16, 108]. Firstly, it does not provide an end-to-end guarantee, which may be required by some applications. This is because *DiffServ* does not keep per-flow state information. Secondly, during congestion, *DiffServ* drops some low-priority packets, and this may waste the resources previously reserved. In addition, *DiffServ* is not designed to cope with node mobility and the dynamically changing network topologies exhibited by MANETs. In MANETs, all the nodes are homogeneous as far as *QoS* provisioning is concerned. Owing to node mobility and the absence of a fixed infrastructure, it is difficult to classify network nodes into core nodes and edge nodes.

The resource reservation-based and service differentiation-based approaches discussed above were originally designed for use in static wired networks. The next two sections are devoted to *QoS* solutions that are specifically designed for MANETs.

2.5 QoS Frameworks

A *QoS* framework refers to a set of software components that inter-work and cooperate to provide end-to-end *QoS* services to an application or data stream [56]. The Flexible *QoS* Model for MANETs (*FQMM*) and *INSIGNIA* are the most notable *QoS* frameworks designed for MANETs.

FQMM [123, 31, 108] is the first *QoS* framework proposed for MANETs. It uses a hybrid approach that integrates the reservation-based and class-based *QoS* services provided by the *IntServ* and *DiffServ* models. It uses the per-flow reservation-based *QoS* service of the *IntServ* model for the high-priority traffic, but the per-class *QoS* service of the *DiffServ* model for the remaining (non-high-priority) traffic. This hybrid approach reduces processing and memory requirements in comparison with the approach taken by the *IntServ* model, thus making it more scalable, while at the same time, providing optimum *QoS* services to high priority traffic. The main drawback of this framework is that it cannot adapt to the dynamic topology changes in MANETs.

INSIGNIA [59, 31, 35, 108, 75] is a *QoS* framework designed for MANETs with per-flow granularity and support for mobility. It supports two levels of *QoS*: resource reservation for high priority traffic and best-effort delivery for low priority traffic. The main goal of *INSIGNIA* is to provide adaptive *QoS* services. It uses *QoS* reports to inform the source node of the success or failure of a reservation. These reports allow the source node to 'scale-up' or 'scale-down' *QoS* requirements in response to measured network conditions. *INSIGNIA* first attempts to support the maximum bandwidth requirement if sufficient resources are available. Otherwise, they scale-down to satisfy the minimum requirement. If there is insufficient bandwidth to support the minimum flow requirement, the priority of the packets in the flow may be further scaled-down to the best-effort delivery service. Packet priorities may be increased if sufficient resources later become available. Similarly, the best-effort delivery can be scaled-up to an enhanced *QoS* when sufficient resources become available. *INSIGNIA* does not include its own routing protocol [75], i.e. it is not coupled to a particular routing protocol. It is designed to work with *AODV* and *DSR*. However, since the flow state information is kept within the nodes in the path, *INSIGNIA* may suffer from scalability problem [122].

2.6 QoS Routing Protocols

QoS routing protocols are designed to satisfy certain *QoS* requirements, e.g. low packet delivery delays and/or high packet delivery ratios, in the presence of limited available bandwidth resources and node mobility [30, 99, 21].

Some of the existing *QoS* routing protocols are extended by adding relevant *QoS* parameters to the routing table or routing packets of traditional routing protocols. This is done to enhance the path selection process, and/or to eliminate the paths that do not support the *QoS* required. These protocols can be largely classified into proactive and reactive *QoS* routing protocols (explained in the next subsections).

There are also some *QoS* routing protocols that focus on achieving a reliable data transmission in the presence of MANET characteristics (discussed in section 1.1). In other words, in these protocols, the focus is on selecting nodes that are reliable in delivering data. Examples of these *QoS* routing protocols are route-break prediction, cluster-based, energy-aware, location-based, and multipath routing protocols (explained in the next subsections).

However, these protocols do not provide *QoS* guarantees. Instead, they improve the average *QoS* offered to all the packets in a given network condition.

2.6.1 Proactive QoS Routing Protocols

The proactive *QoS* routing protocols are designed to satisfy some specified *QoS* requirements by adding relevant *QoS* parameters into the proactive routing protocols. The added *QoS* parameters are used to select an optimal path between the communication nodes. The path selection depends on the corresponding *QoS* parameter values required. For example, $QOLSR$ [9] is an extension of the $OLSR$ protocol [41, 43, 79] that adds delay and bandwidth metrics in each routing table entry corresponding to each destination. These metric values are estimated between each node and its neighbouring nodes. Based on these metric values, an optimal path is sought in terms of bandwidth and delay.

2.6.2 Reactive QoS Routing Protocols

The reactive *QoS* routing protocols are designed by adding the *QoS* parameters into the reactive routing protocols. Quality of Service $AODV$ (*QoS*-$AODV$) and the On-demand *QoS* routing (OQR) protocols are examples of this category of protocols. They are both extended from the $AODV$ [111, 20, 97] protocol. *QoS*-$AODV$ [90] adds the average end-to-end packet delivery delay as the *QoS* requirement and the value of this *QoS* is carried in the $RREQ$ packet header. With OQR [64], $AODV$ is extended to support the estimation of bandwidth as the *QoS* metric. These protocols select the downstream neighbours, that will form the path, based on the *QoS* metrics.

2.6.3 Route-Break Prediction Routing Protocols

The route-break prediction routing protocols are designed to optimise packet delivery success by selecting stable paths that have a high probability of being available for a long time. The protocols employ route-break prediction methods to predict a link break before it actually happens. Base on this prediction, they discover a new path before the old path breaks. In this way, packet loss as caused by path breaks during the data transmission phase can be reduced. Trigger-based Distributed QoS Routing (TDR) and ADaptive QoS Routing ($ADQR$) protocols are examples of this type. TDR [95] uses neighbours' power levels to predict link failures, whereas, in $ADQR$ [42], the signal strength is used to predict link breaks.

2.6.4 Cluster-Based QoS Routing Protocols

Cluster-based QoS routing protocols are designed for large sized networks. Cluster-based routing protocols can be divided into flat and hierarchical. With the flat approach, nodes in a network are grouped into clusters. Each cluster has a cluster head that has information about all the nodes in that cluster. Nodes in one cluster communicate with each other through the cluster head in the cluster. When sending data packets, if the source node and the destination node are in the same cluster, then the cluster head will forward the data packets from the source node to the destination node. Otherwise, the cluster head forwards the packets to another cluster head which is associated to the destination node. The Cluster-Based QoS Routing ($CBQR$) protocol [109, 5] uses flat approach. With the hierarchical approach, levels of clusters is formed. The lowest level uses the flat clusters approach to form clusters. At higher levels, cluster heads are further clustered and each cluster has a cluster head. This approach proposed for MANETs to achieve scalability. The protocol proposed in [49] uses the hierarchical approach.

Cluster-based QoS routing protocols have two advantages. Firstly, they can reduce the storage overhead and save battery power for the nodes in the network. This is because communication nodes are not expected to maintain any tables or to perform any complex processing. They only communicate through their cluster head. Secondly, they can reduce bandwidth consumption in the network. This is because these protocols prevent flooding of control packets through the entire network; control packets are only forwarded among the cluster heads. These advantages make the cluster-based protocols suitable for large sized networks.

However, cluster-based *QoS* routing protocols may not be suitable for MANET for two reasons [124, 14]. Firstly, the cluster formation and maintenance may be difficult to achieve or resource consuming, particularly if the node mobility level is high. This is because, when the node mobility is high, the network topology will change frequently. As a result, there will be need for frequent formation or rearrangement of clusters and this is a resource consuming process. Secondly, as control packets are transmitted through cluster heads, this can cause battery depletion of the cluster heads more quickly than other nodes, leading nodes to power off, which will lead to rearrangement of the clusters and this imposes additional processing overheads.

2.6.5 Energy-Aware Routing Protocols

With an energy-aware *QoS* routing protocol, node energy is used for path selection and maintenance. During a path discovery phase, intermediate nodes are selected to form a path based on their current energy levels. In this way, the selected intermediate nodes will, with a high probability, survive for the duration of a data transmission phase, thus, ensure that a communication, once initiated, will not be disrupted by link breaks, leading to a better *QoS* in terms of packet delivery ratios. The Stability-based *QoS*-capable Ad hoc On-demand Distance Vector protocol (*SQ-AODV*) [118] is an energy-aware routing protocol.

2.6.6 Location-Based QoS Routing Protocols

Location-based *QoS* routing protocols are location-based routing protocols that use *QoS* parameters to enhance a path selection process. The Predictive Location-Based *QoS* Routing (*PLBQR*) and *QoS* Greedy Perimeter Stateless Routing (*QoS-GPSR*) protocols use this approach.

PLBQR [102] is a location-based *QoS* routing protocol that uses a proactive approach. Each node broadcasts its position and resource information periodically. Once the information obtained, each node stores it in its routing table and uses it to compute a path to any other node in the network. It also estimates path breaks and can, if needed, proactively find an alternate path during data transmission phase.

QoS-GPSR [1] is a reactive location-based routing protocol that provides per-flow end-to-end *QoS* guarantees based on the application requirements. In *QoS-GPSR*, the *GPSR* protocol [48] is used to discover paths. Then, each node makes a path selection decision based on the available bandwidth of each path estimated during the path discovery process and the *QoS* requirement of the application.

2.6.7 Multi-Path Routing in MANETs

Multi-path routing protocols are designed to discover multiple path between a given source and destination pair in a single path discovery process. These protocols can largely be classified into two groups based on the purpose the discovered paths are used for. They may be used for load balancing, thus to increase the reliability in *QoS* provisioning or for maximising packet deliveries.

In the first group, multiple path are used in a sequential manner to increase packet delivery reliability. With this approach, one of the discovered paths is used as the primary path, and the others are used as secondary or backup paths. In the event when the primary path fails, data can still be delivered over a secondary path [31]. This multi-path routing approach can support high mobility networks with a low end-to-end packet delivery delay and high packet delivery ratio. This is because data packets can be promptly delivered along the secondary path, that has already been discovered, if the primary path breaks. However, this approach does not support load balancing in the network. The Ad hoc On-demand Multi-path Distance Vector protocol (*AOMDV*) [31, 73], Multi-path Routing Protocol (*MRP*) [30], and *AODV* Backup Routing (*AODV-BR*) [57] use this approach.

The second group of multi-path routing protocols is designed to support load balancing in the network. In this case, packets are delivered using more than one path simultaneously [58]. Distributing the load over multiple path can reduce the possibility of creating congestion along a certain path or speed up power depletion in certain nodes in the network. These can help to reduce end-to-end packet delivery delays as caused by congested nodes or link breaks due to energy depletion. However, these protocols may increase the computational overheads in source nodes, as additional node complexity is required, at the source nodes, to select paths. Examples are *QoS* and Load Balancing *AOMDV* (*QLB-AOMDV*) [113, 23], Energy Aware Load Balancing Multi-path (*EALBM*) [24], and Multi-path source routing (*MSR*) [119] protocols.

Multi-path routing protocols may also be designed to harvest bandwidths over multiple path to support the delivery of packets generated by a single stream. The bandwidth provided by multiple path can be used to deliver high priority traffic, thus achieving *QoS*. As the bandwidth is limited in a wireless network, discovering a single path that satisfies the required bandwidth may not always be possible in MANET. In this case, multiple path can be used simultaneously to route the data. The resulting bandwidth is the aggregate bandwidth of the paths involved. Ticket-based approach [63] and Split Multi-path Routing protocol (SMR) [58] use this approach.

Study results [80, 78] have shown that, in general, using multi-path routing in ad hoc networks can lead to better performance than using single path routing. However, compared to single-path protocols, there are two disadvantages with the use of multi-path routing. The primary disadvantage is computational complexity and overhead. The additional node complexity results from selecting the best path(s) by a source node. Secondly, maintaining multiple path to each destination node requires the use of a larger routing table at each intermediate node, increasing memory requirements.

It is also worth emphasising that none of the above mentioned *QoS* solutions considers the implications of security attacks on *QoS*. They assume that all the nodes in the network are trustworthy and cooperative in routing data packets.

2.7 Security Threats to QoS Routing Protocols

MANETs are vulnerable to a number of security threats; some may have direct implications on *QoS* provisioning. This section discusses the security threats and attacks focusing on those that have direct implications on *QoS*. Security threats and attacks can largely be classified into passive and active attacks [112, 103].

In a passive attack, such as eavesdropping an attacker attempts to gain unauthorised access to data or network structures. This type of attack may not have implications on the routing operations, thus on *QoS*, of the underlying network, but they may be the prerequest for launching an active attack.

Active attacks are typically aimed at discarding data, disrupting protocol operations, and/or preventing other nodes from participating in a networking operation. The attacks could cause disruptions to the underlying network operations and have detrimental effects on *QoS*. Active attacks can be further classified into resource consumption attacks [81] and routing disruption attacks [27].

In a resource consumption attack, such as a jamming attack, an attacker injects packets into the network to consume the resources of other nodes in the network. The attacker may generate control packets frequently, and send them into the network, or forward stale packets to the nodes in the network. The intention of these attacks is to exhaust network bandwidth, consume computational power and/or deplete battery power of other nodes in the network.

In routing disruption attacks, an attacker tries to disrupt routing operations by dropping packets or by forwarding packets along a wrong path. In this thesis, we focus on packet dropping attacks.

Packet-dropping attacks are a type of denial of service attack by which a malicious node discards packets that are expected to be forwarded to its downstream neighbours. This type of attack can target packets at the network layer. They require attacking nodes to be on the path. A typical attack scenario is like this: during a path discovery phase, the attacker acts normally to situate itself on a path between a source node and a destination node. Once the attacker is included in the path, it drops data packets transmitted by the source node. The attacker can use different strategies to drop packets [34]. It may drop all the incoming data packets (in which case, the attack is called a black hole attack [93, 116, 81]), or drop them selectively (called a grey hole attack [93]).

Packet dropping attacks reduce the average network throughput and packet delivery ratios, or disrupt network operations. All of these will have a detrimental effect on *QoS*. In this research, we only consider the effect of packet dropping attacks, as the core function of *QoS* routing is data packet forwarding [27]. Any other attacks are outside the scope of this research. The next section discusses existing efforts carried out to counter packet dropping attacks in MANETs.

2.8 Countermeasures to Packet Dropping Attacks

Routing protocols that are designed to resist packet dropping attacks can largely be classified into three groups based on the approaches used in these protocols. They are trust-based, multi-path, and acknowledge-based routing protocols.

2.8.1 Trust-Based Routing Protocols

In trust-based routing protocols, nodes monitor the behaviours of their neighbouring nodes to estimate their respective trust values. These trust values are taken into account when making routing decisions [105, 33, 128]. In other words, trust values are used to select which nodes should be included in a path to participate in the packet forwarding process. A node with a high packet dropping rate is given a low trust value by its neighbours. Consequently, a node with a low trust value is more likely to be considered as a malicious node and will be excluded from a chosen path.

There are two different trust models proposed in the literature [39], a global trust model and a local trust model. With the global trust model [128], which is also called a reputation-based model, each node in a network needs to know the trust value of every other node in the network. This is typically achieved by nodes exchanging trust-relevant information over the network. Exchanging information on the scale of the whole network can introduce a high level of traffic overhead and can cause congestion in the network. Also, the trust values conveyed may not be truthful; they may be faked or altered during transit. To protect the trust values against attacks will impose additional processing overheads on to the networks' nodes. The protocol proposed in [33] uses the global trust-based model.

In [33], some nodes are selected to be trust authority nodes. These nodes collect any complaint reports from other nodes about their neighbours. The trust authority nodes integrate their direct observations of malicious nodes with the complaint reports they receive from other nodes to create a global reputation vector. This vector will then be distributed to all the nodes in the network. If the trust level of a node drops below a certain threshold, then the node will be classified as a malicious node.

With the local trust model [88, 39], on the other hand, each node only needs to learn the trust values of its immediate neighbours. They learn these values by direct experiences or interactions with, and observations of, their neighbouring nodes. This local trust model has two advantages in comparison with the global trust model. Firstly, the local trust model generates less traffic overhead in the network. Secondly, it imposes less computational overheads onto network nodes, as measures applied to protect the trust values carried in the packets can be avoided. However, this method may not detect packet dropping caused by overloaded nodes [115].

2.8.2 Multi-path Routing Protocols

Packet dropping attacks may be countered by using multiple path routing protocols. There are two approaches taken in the design of these protocols. The first one is to duplicate data packets and send them through multiple path, one primary path, and one or more secondary paths. The secondary or backup path(s) is used to send the duplicated traffic. Packet duplication can increase delivery reliability. For example, if a data packet is duplicated and sent via m paths, then the data packet may still be delivered to the destination node even if $m-1$ paths fail. However, this approach increases the overhead arising from the duplicated data packet transmissions. The Redundant Multi-path Source Routing ($RMSR$) [127], and the Redundant Source Routing (RSR) [120] protocols duplicate traffic along a selected secondary path.

In the second multi-path routing approach a single packet is split into multiple blocks and these multiple blocks are then transmitted over multiple path in parallel. The limitation of this approach is the computational complexity imposed on a destination node as the destination node needs to reassemble the blocks. The Secure Message Transmission (SMT) protocol [86] and the protocol proposed in [114] use this routing approach. They split each packet in such a way that each block of the packet contains some additional bits. The additional bits are calculated in such a manner that the original packet can be reconstructed given a subset of the blocks.

2.8.3 Acknowledgement-Based Routing Protocols

Acknowledgement-based routing protocols use an acknowledgement (ACK) packet to acknowledge the receipt of a data packet. Two types of acknowledgements hop-to-hop or end-to-end acknowledgements can be used to directly locate a link where packets are lost. In the hop-by-hop ACK scheme, when a node forwards a data packet successfully over the next hop, the node at the other end of the next-hop link (i.e. the node two hops away) will send back a two-hop acknowledgment. The acknowledgment packet is used to indicate that the data packet has been received successfully. This will continue until the data packet reaches the destination node. The protocols presented in [129], [12] and [67] use this scheme.

The end-to-end ACK scheme detects packet loss though transmitting ACK packets by the destination node to the source node for each successfully received data packet. The source node estimates a trust value for each link forming the path based on the *ACK* packet received from the destination. If an acknowledged packet is received, the source node increases the trust value for each link along the path to the destination. Otherwise, if no acknowledgment packet is received, the source node considers that the packet is lost and will retransmit it. Every retransmission decreases the trust value for all the links in the path. A link with a low trust level is avoided during path selection. With the end-to-end acknowledgement-based routing scheme, it is hard for the source node to work out which node has actually discarded a packet, so the source node decreases the trust value of the entire set of nodes forming the path. This could be unfair to well behaved nodes that are involved in the path. In addition, sending an *ACK* packet for each received data packet is expensive in term of bandwidth consumption. *Trusted-DSR* [44, 115, 8, 6, 86, 125, 107] uses this routing scheme.

The above protocols are largely designed to address packet dropping attacks, they have not explicitly considered the issue of *QoS*. It is also important to emphasise that none of the above discussed solutions has considered the impact of fluctuating bandwidth as caused by node mobility along with packet dropping attacks.

2.9 Integrated Consideration of QoS and Security

As mentioned earlier, it is essential to consider security with *QoS* as one impacts the other [76]. This section surveys the work on integrating security and *QoS*. The author of [100] proposes a secure *QoS* framework called Cluster-based Routing for End-to-end Security and Quality of service satisfaction (*CRESQ*). *CRESQ* is a cluster based routing protocol which effectively uses the idea of clustering to reduce routing overheads and to provide *QoS* guarantees. The *QoS* guarantees are done through the use of resource reservations during the forwarding of a *RREP* packet. A cluster head forwards the *RREP*, only if it can satisfy the *QoS* and makes reservations on itself. For the implementation of security, each cluster head forwards a *RREQ* and *RREP* packet only to reliable cluster heads. However, in a highly mobile MANET, a resource reservation could be wasted, This is because, when the topology changes, the clusters will need to be rearranged, the assignment of nodes to clusters will change and a reservation made before the cluster rearrangement may become void. If it is void, a new reservation will be required. Repeated reservations will not only introduce additional bandwidth overheads, but also impose additional processing overheads on nodes in the network.

In [70], the authors proposed a secure *QoS* signalling protocol that has three features. Firstly, it reserves resources on the nodes that form the path, to provide the *QoS* required by an application. Secondly, the protocol has a built-in authentication service to authenticate the *QoS* signalling messages that are used to reserve the resources. The authentication is done on a hop-by-hop basis and it uses digital signatures. Thirdly, the protocol can detect malicious attacks on *QoS* parameters by using overhearing techniques. The destination node and intermediate nodes monitor the flows against the promised *QoS* requirements. They periodically send reports to the nodes forming the path including the source node. This work has two limitations. Firstly, it assumes that the network is reliable and resilient against malicious attacks and only provides authenticity protection to *QoS* signalling messages. Secondly, it has not considered the impact of mobility on *QoS*.

The authors in [110], [37] and [104] proposed frameworks that balance between security levels and *QoS* provisioning, afforded to the traffic in a network. These frameworks provide a security level in response to a threat level in such a manner that maintains good *QoS* provisioning. For each threat level and each associated security level, a particular defensive measure can be applied, and a certain level of *QoS* can be achieved. At a low threat level, if the *QoS* requirement cannot be satisfied, the security level can be lowered, allowing a higher level of *QoS* to be satisfied. In contrast, if the network is under a higher level of threat, the framework can switch to a higher level of security protection, and as a result, the *QoS* provided may not satisfy the requirement. These frameworks use cryptographic techniques to provide confidentiality, integrity, and authentication protections. However, these designs have not considered the effects of packet dropping attacks on *QoS*.

In [76], the authors proposed an Integrated Security and *QoS* Routing and Data Communication (*ISQRDC*) framework for MANETs. *ISQRDC* balances between security and *QoS* even when the network resource availability changes. It combines a trust-based authentication scheme and an Ant Colony Optimisation (*ACO*) technique. The trust-based authentication scheme is used to estimate the reputation of a node when selecting a downstream neighbour to form a destination node. *ACO* uses ant-like packets to identify a trustworthy path between the source and the destination nodes. The ant-like packets carry the topological information about the nodes in the network thus facilitating the sharing of information among the nodes to maintain paths. However, the use of the ant-like packets to maintain paths introduces a high level of routing overhead in the network, and the overheads increase as the mobility level increases. Furthermore, the size of the ant-like packets increases dynamically as the packets pass through the network toward the destination node.

In [75], the authors proposed a novel 2-Dimensional Adaptation ARChitecture (*2-DAARC*). The architecture supports two forms of adaptations to optimise *QoS* provisioning. When the threat level in the network is low, bandwidth reservations are used along a single path. When the threat level is high, the architecture uses a duplicated, best-effort data packet delivery service along multiple path to deliver the packets. The architecture switches between the single and multiple path delivery services based on the underlying network conditions. However, the issue of how to discover multiple path with minimum bandwidth costs was not considered in this work.

Table 2.1 summarises the literature review. From the table, it can be seen that there are a number of issues which should be considered in designing a *QoS* solution for MANETs, and none of the existing solutions has considered all of these issues. A new solution is therefore required to optimise the *QoS* provisioning in the presence of packet dropping attacks in a dynamic MANET environment. In the following section, we highlight our vision on proposing such a solution.

2.10 Our Aim and Challenging Issues

Our aim is to find a solution that can best support the delivery of high priority traffic (i.e. to deliver them with as little packet delivery delay, and as high packet delivery success, as possible) and to achieve this with as little communicational cost as possible in the presence of node mobility and packet dropping attacks. To accomplish this, we have four challenging issues [36, 35, 30, 99] to address. These challenging issues are due to MANET characteristics and their impact on *QoS*.

Table 2.1: Literature Review Summary

QoS Solutions	Accommodate Topology Changes	Handle Bandwidth Fluctuations	Reduce Bandwidth Costs in Path Discovery	QoS Provisioning	Reduce Complexity Cost on Nodes	Resilient to Packet Dropping Attacks
Topology-Based	X	X	✓	X	X	X
Location-Based	✓	X	✓	X	✓	X
Resource Reservation	X	X	X	✓	X	X
Service Differentiation	X	X	X	✓	✓	X
QoS Frameworks	✓	✓	X	✓	✓	X
QoS Proactive	X	X	X	X	X	X
QoS Reactive	X	X	✓	X	✓	X
Route-Break Prediction	✓	X	✓	X	✓	X
Cluster-Based	X	X	✓	X	✓	X
Energy-Aware	✓	X	✓	X	✓	X
QoS Location-Based	✓	X	✓	X	✓	X
Multi-Path	✓	✓	✓	X	✓	✓

The first is how to accommodate dynamic topology changes in path discovery. In MANETs, the mobility of the nodes causes link failures and paths breakage, resulting in frequent changes in network topologies, which, in turn, will cause packet losses. The lost packets will need to be retransmitted, and the repeated retransmissions can lead to network congestion further increasing packet delivery delays and losses. All of these will have detrimental effects on *QoS*. In other words, the challenging issue here is how to find paths that are the most reliable set to deliver high priority traffic so as to satisfy their *QoS* requirements with minimum overhead costs, and how to reduce the effects of the link breaks on *QoS* performance.

The second issue is how to cope with fluctuating available bandwidth. MANETs usually have limited and fluctuating bandwidth due to frequent network topological changes. It is not always possible to find a single path that is both reliable and has the sufficient bandwidth to support the delivery of a high-priority traffic flow. There is clearly a need for making use of bandwidths provided by two or more paths. So, the challenging issue here is how to make the best use of the fluctuating bandwidth to serve *QoS* efficiently and effectively under various network conditions, e.g. different levels of network traffic loads and different levels of node mobility.

The third issue is how to optimise *QoS* provisioning without the support of central authority and without imposing (or minimising) trust on intermediate nodes. A MANET relies on collaboration among communication nodes in the network to route traffic for each flow. Some communication nodes may be malicious. They may launch black hole or grey hole attacks. Therefore, how to estimate the trust level of a node, how to find more trustworthy paths to deliver high priority traffic, and how to do so with minimum overhead cost is also a challenging issue.

The fourth issue is that resources of MANET nodes are typically more limited compared to wired network nodes in terms of storage and processing capabilities and also they are battery energy [118]. Excessive node complexity can consume resources and deplete battery power more quickly, leading nodes to power off, and this will adversely affect *QoS*. Therefore, it is important to address the above mentioned challenging issues with as little complexity as possible, especially on intermediate nodes

2.11 The Way Forward

Based on the review of the related literature and the challenging issues identified in the previous sections the following ideas are generated for a new solution to achieve *QoS* in MANETs containing packet forwarding attackers. These ideas include the design of an adaptive solution which combines a single path routing service with a multi-path routing service. The adaptation is necessary to accommodate the dynamic MANET features as caused by mobility, fluctuating available bandwidth, and attacker ratios and to optimise high priority packet deliveries. When a single path cannot satisfy the bandwidth requirement of a traffic flow, multiple path should be used. To reduce the overheads in multiple path discoveries, we will use a location-based approach to limit the number of control packets poured into the network. To increase or optimise delivery success, we should use more trustworthy paths as much as possible, and the selection of more trustworthy paths should not impose much overhead costs.

2.12 Chapter Summary

This chapter has surveyed and discussed the related works in the literature in the area of *QoS* provisioning in MANETs. It has also explained the security threats and attacks on *QoS* routing protocols. Based on this literature survey, some new ideas to achieve *QoS* in a malicious MANET environment have been proposed.

The next chapter presents the building blocks used to implement the novel ideas, and the evaluation methodology used to evaluate the implementation.

Chapter 3

ATL-QoS Framework Design

3.1 Chapter Introduction

The *ATL-QoS* framework intends to optimise *QoS* provisioning in MANETs containing packet dropping attackers. It is a network-layer solution comprising two planes: a control plane and a data plane. The control plane is responsible for multiple path discovery, the selection of a set of more trustworthy paths from the discovered paths, the allocation of traffic to the selected paths and path quality verification to address packet dropping attacks. The data plane is responsible for packet transmission, forwarding and reception. The major novel contributions presented in this thesis are all in the control plane, i.e. two novel protocols for multiple path discoveries (presented in detail in Chapters 5 and 6, respectively), and a novel solution for path selection, traffic allocation and path verification for high priority traffic delivery (presented in detail in Chapter 7). To evaluate these novel solutions under various network conditions, they are implemented with all the supporting components in the control plane as well as the functions in the data plane. The resulting implementation is our *ATL-QoS* framework.

This chapter describes the *ATL-QoS* framework. Section 3.2 specifies the high-level requirements for the design of an efficient *QoS* solution for MANETs containing packet dropping attackers, along with the idea or measures taken to realise the requirements. Section 3.3 gives design preliminaries, the assumptions, definitions used in the design and the notation used in describing the design. Section 3.4 describes the architecture of the framework and the architectural components are described in Section 3.5. Section 3.6 gives a run through of the operations of the framework, and finally Section 3.7 summarises the chapter.

3.2 Design Requirements and Measures Taken

In addressing the design challenges in providing *QoS* for MANETs (identified in Chapter 2), seven requirements are specified.

R 1: The solution should optimise the provision of *QoS* services in terms of packet delivery delays and packet delivery ratios.

R 2: The solution should accommodate dynamic topology changes.

R 3: The solution should accommodate fluctuating available bandwidth.

R 4: The solution should be resilient to packet dropping attacks.

R 5: The solution should not place trust on intermediate nodes.

R 6: The solution should reduce computational cost on the intermediate nodes.

R 7: The solution should reduce bandwidth cost introduced into the network.

To satisfy the requirements specified above, we present a framework called an Adaptive Trust-Aware Location-based (*ATL-QoS*) framework. The design of the framework has taken the following five measures to address the identified challenging issues while satisfying the requirements. In the following measures, there may be conflicts between the requirements, and some compromise between them will be needed.

Measure 1: The framework should be able to discover multiple path between a pair of nodes (source and destination) and while doing so, the bandwidth costs incurred should be as low as possible. Usually, discovering as many paths as possible requires broadcasting path discovery packets in the whole network. This process is expensive in terms of bandwidth cost. We need to reduce the size of the search area where the path discovery packets are broadcast, however, this may lead to a reduction in the number of paths discovered. In other words, there is a trade-off between the routing overhead incurred in discovering multiple path and the number of paths that can be discovered. To optimise this trade-off, the *ATL-QoS* framework design employs two ideas. The first is to use a directional approach to multiple path discovery. With this approach, path discovery packets are forwarded within a search area bounded by the source and destination nodes, rather than flooding the

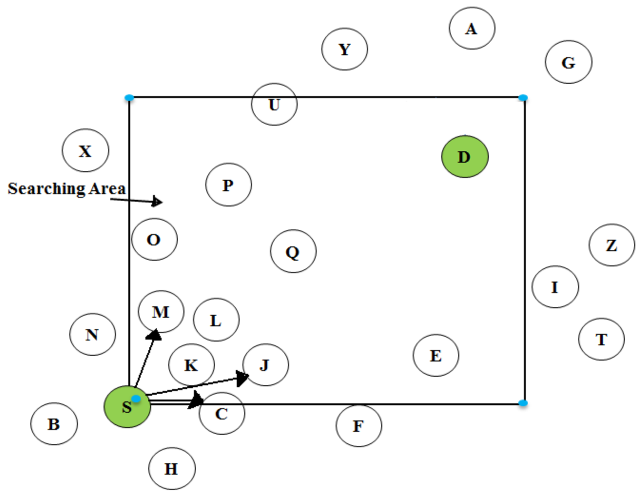


Figure 3.1: Using Directional Approach and Subset of Neighbouring Nodes

whole network regardless of the locations of the two nodes. The second idea is for a node only to send the path discovery packets to a subset of its neighbouring nodes governed by the trust level of the neighbouring nodes, rather than broadcast the path discovery packets to all the nodes in the search area. Using these two ideas, we can reduce the number of control packets of broadcast traffic injected into the network, thus reducing the bandwidth cost introduced (satisfies requirement R7).

Figure 3.1 uses an example to illustrate the use of directional approach and subset of neighbouring nodes. As shown, to reduce the number of control packets, the source node, N_S, send a *RREQ* packet to a selected set of downstream nodes. N_S has eight neighbouring nodes N_B, N_N, N_M, N_L, N_K, N_J, N_C and N_H. The nodes N_N, N_B and N_H will be excluded as they are not in the direction (search area) of N_D. N_S selects the set of neighbouring nodes from the neighbours that are in the search area of N_D based on their trust value. In this example, N_M, N_J and N_C have the highest trust value and they are chosen.

Measure 2: The multiple path discovery function provided by the framework should be able to adapt to the underlying network conditions. In other words, it should be adjustable in adaptation to the underlying network conditions. For example, if attacker ratio in the network is high, increasing the search area may increase the possibility of finding more trustworthy paths. Alternatively, if the attacker ratio in the network is low, we may be able to find enough trustworthy paths in a smaller search area, and smaller search area means less control overhead is generated in the network, which will be beneficial to *QoS* provisioning. Similarly, if the traffic flow requires a lower level of bandwidth, then a smaller number of paths may be sufficient, and in this case, a smaller search area may be used. This indicates that by dynamically adjusting the search area in response to the underlying network condition captured by network parameter values, such as neighbouring node attacker ratios, neighbouring node mobility level, the time of the previous search attempt and the bandwidth required by the traffic flow; we may be able to optimise *QoS* support. Adjusting the search area size can satisfy requirements R2 and R7.

Measure 3: The framework should be able to select the most reliable nodes to form the paths so as to optimise packet delivery ratios during the data transmission phase, and do so with as little communication overhead and computational complexity imposed on the intermediate nodes as possible. For this, we have defined and introduced the use of node and link trust values such that *ATL-QoS* uses a node's trust values to select its neighbours in path formation and uses link trust values to select the most reliable paths for high priority traffic delivery. As mentioned in Chapter 2, there are two different trust models proposed in the literature, a global trust model and a local (direct) trust model. The global trust model, might be used to provide more trustworthy estimations about node and link trust values. However, with global trust model, nodes exchanging trust-relevant information over the network. Exchanging information on the scale of the whole network can introduce a high level of traffic overhead and can cause congestion in the network. Also, the trust values conveyed may not be truthful; they may be faked or altered during transit. Protecting the trust values against attacks will impose additional processing overheads on to the networks' nodes. To reduce the overhead incurred in trust estimation, we have chosen a direct trust model. This measure is taken to satisfy R1, R4, R6, and R7.

Measure 4: To optimise packet delivery of high priority traffic, the framework should be able to deliver the traffic over multiple path, and do so adaptively in response to the bandwidth requirements, available bandwidth and attacker ratio. Three ideas are used to implement this measure. The first is to differentiate the delivery of traffic with different priorities. Two priorities of traffic are recognised: low priority and high priority. The low priority traffic is transferred using the Single Path delivery Service (SPS), whereas the high priority traffic is delivered using one of the two services, SPS or Multiple Path delivery Service (MPS). If a single most trustworthy (i.e. the path discovered with the highest trust value) path can satisfy the required bandwidth, then this path is used to transfer the traffic. In this case, we say SPS is used. Otherwise, if no single path with sufficient trust value could be found and/or it alone could satisfy the required bandwidth, a subset of paths that together can satisfy the bandwidth required is selected. In this case, we say MPS is used. The second idea is to allocate the data traffic on the selected paths based on their respective available bandwidths, and trust values. In this way, we may achieve both load balancing, and optimise successful packet delivery. The third idea is to duplicate the traffic on two independent paths if the path has low trust values, so that in an event where there is a shortage of trustworthy paths, we can make use of paths that are not sufficiently trustworthy, but offset the probability of packet loss by duplicating transmissions over these paths. This is to make the best use of network available bandwidth to serve high priority traffic deliveries, thus satisfying R1 and R3.

Measure 5: The available bandwidth of a path may fluctuate due to node mobility. In addition, as *ATL-QoS* assumes that intermediate nodes along a path are not trustworthy, the available bandwidth and the trust values of the selected paths should be further verified, as an intermediate node could be a packet dropping attacker. If there is such an attacker, the estimated bandwidth and trust value will not be its true values. In the *ATL-QoS* framework, a source node uses feedback provided by the destination node to verify packet delivery ratios along the paths. The destination node feedback represents the number of data packets successfully received at the destination node through that path. The source node uses this feedback to verify the path's available bandwidth estimated during the previous path discovery phase. If an intermediate node has lied about the available bandwidth, the source node may make an incorrect decision in path selection. The feedback from the destination node would allow the source node to adjust this decision after a round trip delay. In other words, any false claim of the available bandwidth along the path can be detected once the destination feedback is received. This measure is taken to satisfy requirements R4 and R5.

Measures 1, 2 and 3 are implemented in Paths Discovery components in Section 3.5.4 employed in the $TADL$ and $TADLV2$ protocols described in Chapter 5 and Chapter 6, respectively. Measure 4 is implemented in Path Selection and Traffic Allocation component described in Section 3.5.6 which is employed in the SAV solution, detailed in Chapter 7. Measure 5 is implemented in Path Quality Verification component described in Section 3.5.6 and also employed in the SAV solution, detailed in Chapter 7.

3.3 Design Preliminaries

This section gives the assumptions, definitions and notations used in the *ATL-QoS* framework design.

3.3.1 Assumptions

The following assumptions are used in the *ATL-QoS* framework design.

A. 1 Properties, including the trust value and bandwidth, are associated with the link; the properties of two links in opposite directions between two nodes may not be identical. Each node may have a different prosperities assigned to each link. For example, for N_A, the trust value of N_B (or the trust value for the channel from N_A to N_B) may not be the same as the trust value of N_B for N_A, i.e. the link from N_B to N_A.

A. 2 Packet dropping are the only security threat considered in this thesis. Packet dropping can be caused by malicious or non-malicious events. We define a malicious packet dropping to be attacks that are only mounted on data packets. These attacks do not discriminate between priority and non-priority data packets. In other words, it is assumed that attackers drop data packets, regardless of their priority types; they forward control packets. The justification for this assumption is that to address packet dropping attacks, attackers must be included in a path, so they must participate in path discovery phase. However, non-malicious events, such as buffer overflow and selfish nodes may also lead to packets lost, but packet loss caused by non-malicious events would happen to both data and control packets. We classify a node with buffer overflow as an overloaded node, and an overloaded node drops both data and control packets. The selfish node behaviour is similar to the behaviour exhibited by an overloaded node. When a selfish node receives a data or a control packet, it simply drops it. They aim to utilize its limited resources only for their own purpose to save their resources to the maximum.

A. 3 The neighbouring node's attacker ratio, that is the ratio of the number of neighbouring nodes who are packet dropping attackers over the total number of neighbouring nodes, is assumed to be known to all the nodes. This is because we did not have time within the scope of the project time to conclude a way to calculate it.

A. 4 Source and destination nodes trust each other. Threats are only from intermediate nodes.

A. 5 It is assumed that the bandwidth requirement is sent to the network level from the application layer in packets/s.

A. 6 Based on the node functionality, there are three types of nodes in the network, these are source node functions, destination node functions, and intermediate node functions. Every node in the network should have all three functions, source node, destination node, and intermediate node, but for simplicity and without losing generality, we implement different functions in different nodes, i.e. source nodes only perform the source node functions, and so on.

3.3.2 Definitions

The framework design makes use of a number of quantifiers or concepts. These are trust value, available bandwidth, path quality, QoS requirement, a node's predicted location, and search area. Below we give the definitions of these quantifies.

Trust value: Trust values are used in the selection of neighbours and paths. Trust here means the reliability of a link, a path or a set of paths in delivering packets. There is a trust value for each link and for each path. The trust value for the link from N_j to N_i reflects the reliability of the link linking N_j and

N_i. Reliability is measured in terms of the ratio of packets that are successfully delivered to their intended destination. Each node estimates a trust value for each of its neighbouring nodes based on the reliability of the link connecting to the neighbouring node. A link trust value is measured in a given range with a minimum and maximum trust value. Each path also has a trust value. The path trust value is calculated based on the trust value of the links forming the path. The set of paths trust value is calculated based on the trust value of the paths forming the set. The method for links' trust value estimation is explained in detail in Chapter 4, whereas, the methods for estimating the paths and sets trust value are explained in detail in Chapter 7.

Available bandwidth: Similar to trust values, each link has a link available bandwidth, and each path has a path available bandwidth. The available bandwidth for a path is calculated based on the available bandwidths of the links forming the path. Path available bandwidths are used for path selection and traffic allocation to the selected paths. The value of the link available bandwidth reflects the maximum achievable transmission rate of the link linking two nodes in a given time period. Each node estimates a link available bandwidth for each of its neighbouring nodes by using the method explained in Chapter 4, whereas, estimating path available bandwidth method is explained in Chapter 7.

Path quality: Path quality (i.e. the quality of the path) is estimated based on the path available bandwidth and path trust value. These values are estimated based on the weakest link principle. The estimation method is explained in Chapter 7.

QoS requirement: The *QoS* requirement mentioned in this thesis refers to a bandwidth requirement at the network level. It is used to support the delivery of high priority traffic (i.e. to deliver them with as little packet delivery delay, and as high packet delivery success, as possible).

A node's predicting location: The predicted location of a destination node, N_D, is an area where source node, N_S, expects N_D to be in at time T_{cur}, based on the known N_D's physical location and speed at time T_{lst}^{SD}. The method used to predict a node location is explained in Chapter 5.

Searching area: This is an area where path discovery packets are forwarded. In the *ATL-QoS* framework, to reduce control overheads, the control packets are forwarded in a specific area, called search area, in the direction of the destination node, instead of broadcasting them in the whole network. The search area is a rectangular area that starts from the current location of the source node, N_S, and includes the predicted location of the destination node, N_D, at the current time T_{cur}. The algorithm for calculating the search area is described in Chapter 5.

Neighbouring nodes' attacker ratios: This is the ratio of the number of neighbouring nodes who are packet dropping attackers over the total number of neighbouring nodes.

Neighbouring nodes' average mobility level: This is the average roaming speed of the neighbouring nodes.

3.3.3 Notations

The notations used in the description of the *ATL-QoS* design are summarised in Table 3.1. These notations are then used throughout the thesis.

3.4 The Architecture

This section describes the architecture of the *ATL-QoS* framework. Figure 3.2 shows the main functional blocks and the interactions among them. The architecture has two main parts, a control plane part, and a data plane part. They are described in the following subsections.

3.4.1 The Control Plane

The control plane is the core of the architecture. It is largely responsible for making routing decisions, i.e. discovering and selecting paths to forward data packets. The structure of the control plane is of three types based on the functions performed by a node. As shown in Figure 3.3, these are source node functions, destination node functions, and intermediate node functions. In real life, a MANET node is expected to perform all the three functions. That is, each node is expected to send data to other nodes, receive data from other nodes, and function as a router (intermediate node) to route packets for other nodes. For clarity without losing generality, hereafter, we assume that each of the functions is performed by a distinct group of nodes. In other words, source nodes perform source node function, intermediate nodes perform the intermediate node function, and the destination nodes perform the destination node function.

Table 3.1: Notations Used in the *ATL-QoS* Design Description

Notation	Definitions	Data Type
N_S	Source node	IP address
N_D	Destination node	IP address
N_i	Network node i	IP address
V_i	N_i mobility speed	Float
(X_i, Y_i)	Physical location of N_i in the x and y coordinate respectively	Float
P_x	The x^{th} path connecting between a pair of source and destination nodes	-
L_{ji}	The link connecting N_j to its neighbour N_i	-
T_{cur}	The current instance of time	Float
T_{cur-1}	Previous instance of time	Float
T_{str}	Starting counting time	Float
T	A period of time from T_{cur-1} to T_{cur}	Float
T^{ji}_{lst}	The last time N_j heard about N_i	Float
T_{trn}	Data packet transmission time	Float
T_{RREQ}	The arrival time of the $RREQ$	Float
TRV	TRust Value	Float
$TRV_{L_{ji}}$	Trust value for the link L_{ji} linking N_j to N_i	Float
TRV_{min}	Minimum trust value	Float
TRV_{max}	Maximum trust value	Float
TRV_{P_x}	Trust value for the path P_x	Float
ABW	Available BandWidth	Integer
ABW_{Lji}	Available bandwidth of the link L_{ji}	Integer
ABW_{P_x}	Available bandwidth of the path P_x	Integer
BW_{req}	The bandwidth required by an application	Integer
SAS	Searching Area Size	Integer
SAS_{exs}	Existing search area size	Integer
CTR	ConTRol packets namely, ($RREQ$, $RREP$, $RERR$, ACK, and DFB)	Integer
$CTR_{suc,Lji}$	Number of successfully delivered control packets over the link L_{ji}	Integer
$CTR_{all,Lji}$	Number of control packets sent over the link L_{ji}	Integer
DAT	data packets	Integer
$DAT_{suc,Lji}$	Number of successfully delivered data packets over the link L_{ji}	Integer
$DAT_{all,Lji}$	Number of data packets sent over the link L_{ji}	Integer
$DAT^{SD,T}_{suc,Px}$	Number of successfully received data packets at N_D via P_x during T	Integer
$DAT^{SD,T}_{snt,Px}$	Number of sent data packets from N_S to N_D via P_x during T	Integer
$HELLO_{suc,Lji}$	Number of $HELLO$ packets successfully transmitted over the link L_{ji}	Integer
$HELLO^{T}_{all,Lji}$	Number of $HELLO$ packet sent over the link L_{ji}	Integer
$RDTimer$	Route discovery timer	Float
$RREQ_{seq}$	$RREQ$ sequence number	Integer

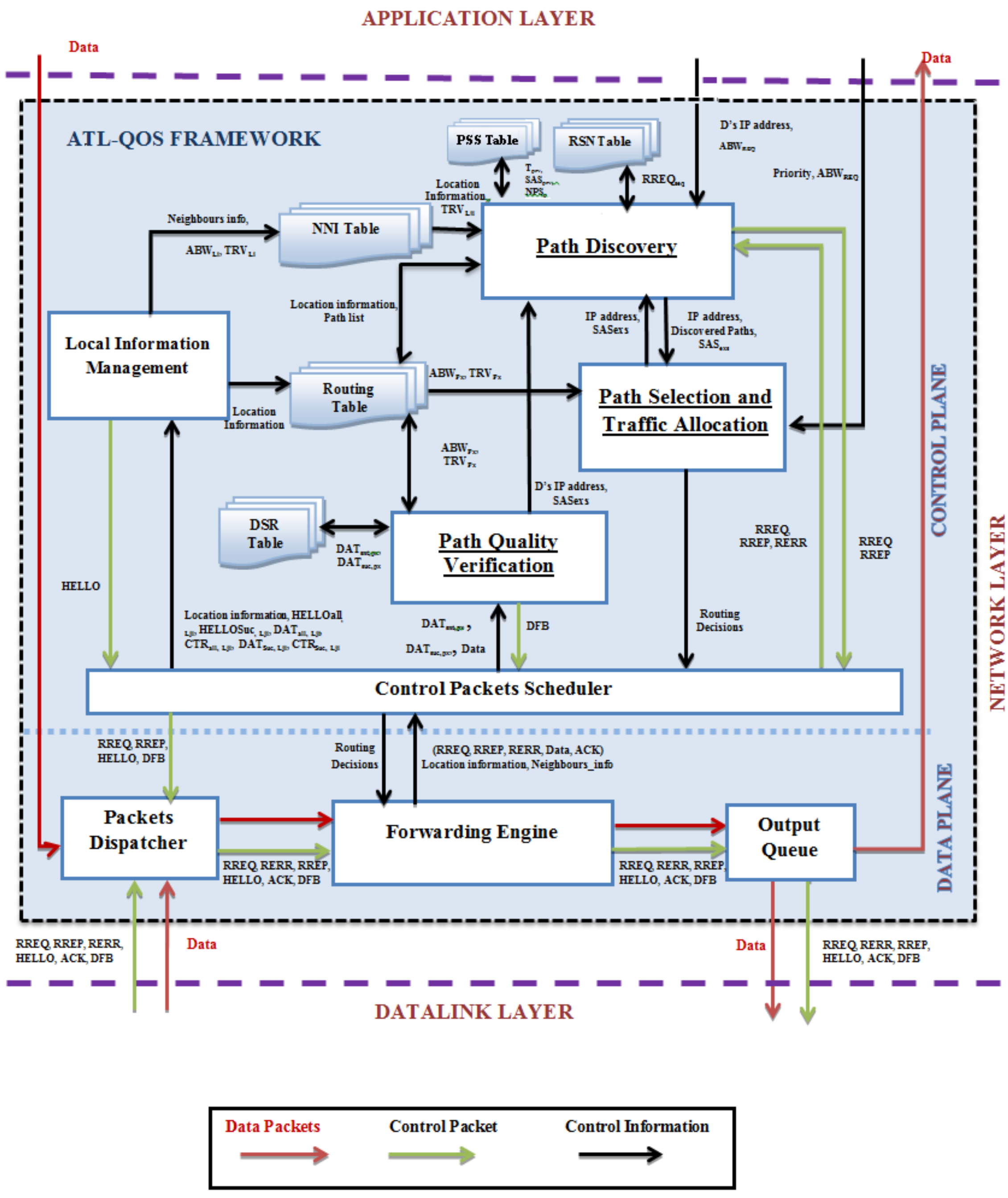

APPLICATION LAYER
Data
Data
ATL-QOS FRAMEWORK
PSS Table
RSN Table
D's IP address, ABW_REQ
Priority, ABW_REQ
Location Information, TRV_Li
RREQ
Neighbours info, ABW_Li, TRV_Li
NNI Table
Path Discovery
Location information, Path list
IP address, SASexs
IP address, Discovered Paths, SAS
Local Information Management
Routing Table
Location Information
ABW_Px, TRV_Px
Path Selection and Traffic Allocation
ABW_Px, TRV_Px
D's IP address, SASexs
DSR Table
Path Quality Verification
RREQ, RREP, RERR
RREQ RREP
CONTROL PLANE
NETWORK LAYER
HELLO
Location information, HELLOall, HELLOSuc, DAT_all, CTR_all, DAT_Suc, CTR_Suc
DAT, DAT, Data
DFB
Routing Decisions
Control Packets Scheduler
RREQ, RREP, HELLO, DFB
Routing Decisions
(RREQ, RREP, RERR, Data, ACK) Location information, Neighbours_info
Packets Dispatcher
Forwarding Engine
Output Queue
RREQ, RERR, RREP, HELLO, ACK, DFB
RREQ, RERR, RREP, HELLO, ACK, DFB
DATA PLANE
RREQ, RREP, RERR, HELLO, ACK, DFB
Data
Data
RREQ, RERR, RREP, HELLO, ACK, DFB
DATALINK LAYER
Data Packets
Control Packet
Control Information

Figure 3.2: *ATL-QoS* Framework Architecture

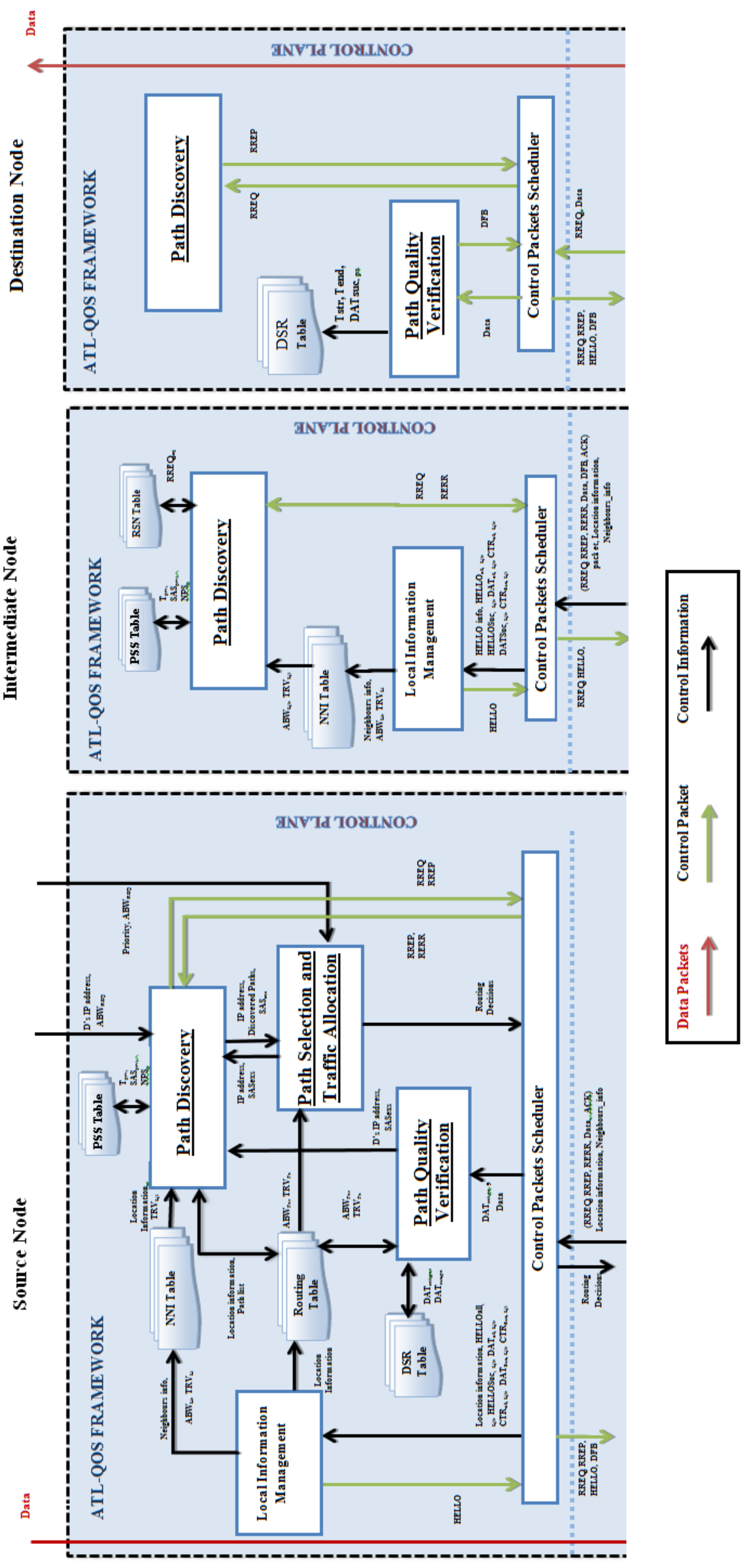


Figure 3.3: *ATL-QoS* Framework Architecture for Source Node, Intermediate Node, and Destination Node Functions, Respectively

Source node functions: A source node generates traffic, discovers a path or a set of paths and sends the traffic to a destination node via the selected path(s). The control plane for the source node contains five functional blocks (components), namely, Local Information Management, Path Discovery, Path Selection and Traffic Allocation, Path Quality Verification, and Control Packet Scheduler. The Local Information Management component of a node is responsible for discovering and maintaining information about the node's neighbouring nodes, estimating their respective link qualities including bandwidths and trust values, and location information. The Path Discovery interacts with the application layer to obtain the bandwidth requirements of an application and discovers paths leading to the destination node. Then, the Path Selection and Traffic Allocation component selects a path or a set of paths (depending on the bandwidth requirement of the application), and allocates the packets to the selected paths for forwarding. During the data transmission phase, the Path Quality Verification component uses feedback from the destination node to adjust the path quality estimated during the path discovery phase. The Control Packet Scheduler gets routing related information from/to the data plane and sends it to/from the other control plane components. These components are described in detailed in Section 3.5.

Intermediate nodes: Intermediate nodes are connected to each other from the source node to the destination node to form a path. The intermediate nodes in the path forward the packets (data and control packets) to and from the destination node. The control plane architecture for intermediate nodes only contains Path Discovery, Local Information Management, and Control Packet Scheduler components.

Destination nodes: A destination node receives packets sent by a source node and sends feedback packets back to the source node. The control plane architecture for a destination node only contains the Path Discovery, path verification, and Control Packet Scheduler components.

3.4.2 The Data Plane

The data plane is responsible for queuing, scheduling and forwarding packets over the selected path(s). It is also responsible for packet reception. The data plane contains three main components; they are a Packet Dispatcher, Forwarding Engine and Output Queue. Upon the arrival of a packet at the Packet Dispatcher, the Forwarding Engine needs to obtain the path(s) from the control plane and forwards the packet toward the destination node. If multiple packets need to be forwarded at the same time, the packets are queued and scheduled at the Output Queue. The architectural components are further described in Section 3.5.

3.5 The Architecture Components

This section describes the *ATL-QoS* framework architectural components and the interactions among them.

3.5.1 The Tables

Each node maintains five tables that are used during path discovery and data transmission phases. These are a routing table, a Neighbouring Node Information (NNI) table, a $RREQ$ Sequence Number (RSN) table, Data packets Received and Sent (DRS) table, and Previous Searching area Size (PSS) table.

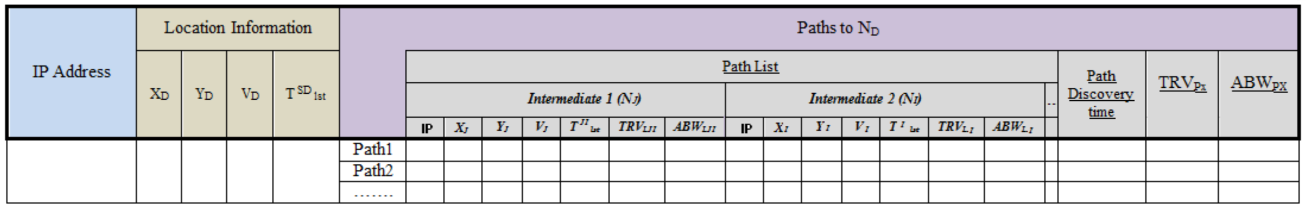
3.5.1.1 Routing Table

Each node in the network maintains a routing table that contains the location and paths information of the discovered nodes in the network. The location information is used to estimate the search area for the corresponding node and the path information is used to determine how to forward data and control packets to the node. In other words, for every discovered node in the network, there is an entry in the routing table. As shown in Figure 3.4, each entry contains the node's unique IP address, location information and path(s) to this node.

The nodes' unique IP addresses have been used, the *ATL-QoS* design is at the network level of the OSI reference model, and the IP address is the most common used type of network level addresses.

For each node, N_D, the location information is captured using four attributes, N_D, physical location (X_D, Y_D), mobility speed (V_D), and the time when N_D, was discovered/updated (T^{SD}_{lst}).

- N_D's physical location (X_D, Y_D): A node N_D's physical location is captured by using X_D and Y_D, where X_D is the node's physical location in the x coordinate and Y_D is the node's physical location in the y coordinate.
- N_D's mobility speed (V_D): This is N_D's mobility speed at the discovered/updated time. It is measured in metres/s.



IP Address	Location Information				Paths to N_D																		
	X_D	Y_D	V_D	T^{SD}_{lst}		Path List															Path Discovery time	TRV_{Px}	ABW_{PX}
						Intermediate 1 (N_J)							*Intermediate 2 (N_I)*							..			
						IP	X_J	Y_J	V_J	T^{JI}_{lst}	TRV_{LJI}	ABW_{LJI}	IP	X_I	Y_I	V_I	T^{I}_{lst}	TRV_{LI}	ABW_{LI}				
					Path1																		
					Path2																		
					…….																		

Figure 3.4: Routing Table

- The time when N_D was added/updated (T_{lst}^{SD}): This is the latest time that N_S heard about the N_D. T_{lst}^{SD} is used to ensure that the entry for this node in the routing table is up-to-date. In other words, if the routing table entry for this node is not updated within a predefined period of time, this entry will be deleted.

Paths to node N_D is used to maintain all the discovered paths to N_D. *ATL-QoS* employs a multiple path discovery protocol, so each entry in the routing table may contain more than one path. Each path is captured using the following attributes:

- A path list: This list may contain multiple entries (one for each path), and for each entry, it maintains information about all the intermediate nodes within the path to N_D. As *ATL-QoS* is a source routing protocol, the source node needs to maintain the whole path up to the destination node in its routing table. The path is then added into the packet header of each outgoing data packet to N_D. Each intermediate node entry contains the intermediate node's IP address, location information, link's (links this intermediate node to the next node in the path) trust value, ($TRV_{L_{ji}}$), and link's available bandwidth ($ABW_{L_{ji}}$). $TRV_{L_{ji}}$ and $ABW_{L_{ji}}$ are, respectively, the trust value and available bandwidth of the link that connect the intermediate node N_i to its neighbouring nodes N_j. These values are regularly estimated by each directly connected node and then distributed to other nodes in the path via control packets during the path discovery phase. During a path discovery phase, when a *RREQ* packet passes through an intermediate node, the intermediate node records these values arrived in the header of the *RREQ* packet. When the destination node receives the *RREQ* packet, its updates the path list and sends it to the source node in a *RREP* packet. The source node needs the path list information to forward data packets during the data transmission phase.

- The path discovery time: This is the time when this node discovers this path. It is used to ensure that the paths stored in the table are up-to-date. Each path is associated to a validity period, if the validity period expires, the path will be deleted from the table, and only up-to-date paths will be stored and used.

- The path trust value, TRV_{P_x}: This is the trust value of a discovered path, P_x. As mentioned earlier, there is a trust value for each discovered path, and this value is estimated based on the weakest link principle. It is the minimum trust value of the links forming P_x.
- The path available bandwidth, ABW_{P_x}: This is the available bandwidth of a discovered path, P_x. Each path has an available bandwidth associated with it, and it is the minimum available bandwidth of the links forming P_x. This is also called the bottleneck bandwidth of the path.

3.5.1.2 Neighbouring Nodes Information (NNI) Table

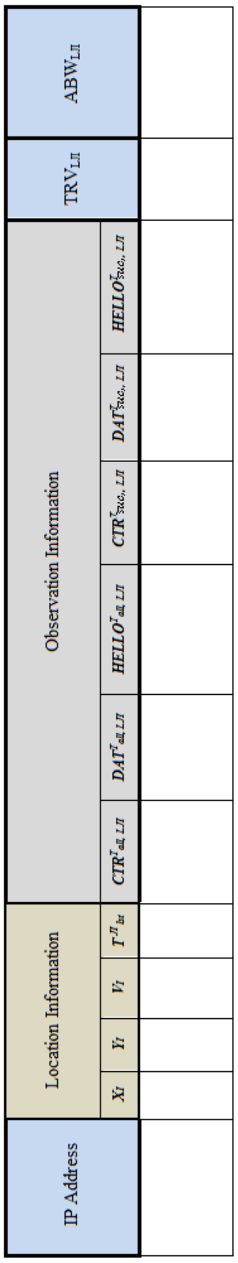
In addition to the routing table, each node in the network also maintains a NNI table storing the information related to each of its directly connected neighbours. The directly connected neighbours are discovered using the neighbour node discovery method, described in Section 3.5.3.2. As shown in Figure 3.5, the neighbour node information is captured using a number of attributes, namely, a neighbouring node's unique IP address, location information, observation information, link's trust value, $TRV_{L_{ji}}$, and link's available bandwidth, $ABW_{L_{ji}}$.

The attributes used to capture a *neighbouring node's location information* have been described earlier in this section, and the values for the attributes are acquired by using $HELLO$ packets, which will be described in Section 3.5.3.1.

The observation information attributes are maintained for each neighbouring node and used to estimate the trust value and available bandwidth of the link connecting this node to each of its neighbours. These attributes are respectively the numbers of control, data, and $HELLO$ packets sent from this node to each of the neighbouring nodes and the number of successfully received control, data, and $HELLO$ packets.

When N_j sends a control, data or $HELLO$ packet to N_i, N_i will acknowledge the receipt of the packet. N_j records the total number of transmitted control and data packets by N_j to N_i during an interval T as $CTR^T_{all,Lji}$, $DAT^T_{all,Lji}$ and $HELLO^T_{all,Lji}$, respectively, and the total numbers of control, data and $HELLO$ packets that are sent by N_j to N_i during the interval T (from T_{cur-1} to T_{cur}) and are successfully acknowledged by N_i as $CTR^T_{suc,Lji}$, $DAT^T_{suc,Lji}$ and $HELLO^T_{suc,Lji}$, respectively.

There is a *trust value, $TRV_{L_{ji}}$*, and an *available bandwidth, $TRV_{L_{ji}}$*, for each link connecting N_j, to each of its neighbours, N_i. These values are estimated



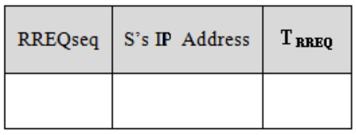
IP Address	Location Information				Observation Information						TRV_{IJ}	ABW_{IJ}
	X_I	Y_I	V_I	T^{IJ}_{Int}	$CTR^I_{all,IJ}$	$DAT^I_{all,IJ}$	$HELLO^I_{all,IJ}$	$CTR^I_{suc,IJ}$	$DAT^I_{suc,IJ}$	$HELLO^I_{suc,IJ}$		

Figure 3.5: Neighbouring Nodes Information (NNI) Table

by using the TVE and ABE methods described in Section 4.3 and Section 4.4, respectively.

3.5.1.3 RREQ Sequence Number (RSN) Table

The third table maintained is a RSN table, shown in Figure 3.6, this records the sequence numbers of all the $RREQ$ packets, $RREQ_{seq}$, that have been forwarded by this node. The table also records the IP address of the source node that has generated the $RREQ$ and the arrival time of the $RREQ$, T_{RREQ}. The pair, $RREQ_{seq}$ and source IP address, is used to distinguish the $RREQs$ generated by different source nodes. T_{RREQ} is used to delete the old entry in the table. In other words, if the T_{RREQ} value for this entry exceeded a predefined period of time, this entry will be deleted.

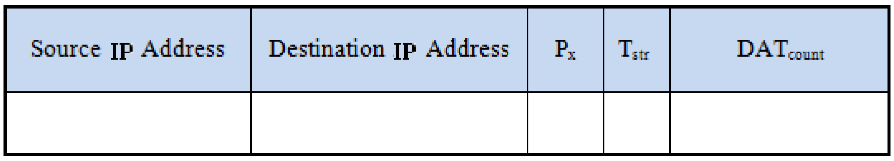


RREQseq	S's IP Address	T_{RREQ}

Figure 3.6: $RREQ$ Sequence Number (RSN) Table

3.5.1.4 Data Packets Received and Sent (DRS) Table

The DRS table is maintained in each source and destination node in the network shown in Figure 3.7. The source and destination nodes use this table during the data transmission phase to record the number of data packets that have been sent from the source node or received by the destination node. The source node and the destination node record the data packets sent or received via P_x during the period T, $DAT^{SD,T}_{snt,Px}$ or $DAT^{SD,T}_{suc,Px}$ respectively in the DAT_{count} field. This table also records the IP addresses of the communicated source and destination nodes and the starting counting time, T_{str}.

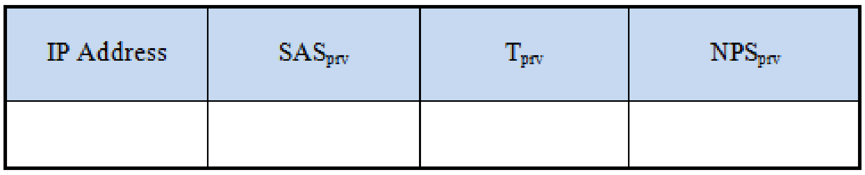


Source IP Address	Destination IP Address	P_x	T_{str}	DAT_{count}

Figure 3.7: Data Packets Received and Sent (DRS) Table

3.5.1.5 Previous Searching Area Size (PSS) Table

The last table maintained is a PSS table. The source and intermediate nodes use this table during the path discovery phase to record information about the previous search attempt. This information will help to determine the starting search area size of the current path discovery attempt. As shown in Figure 3.8, it records IP addresses of the destination node, the last search area size, SAS_{prv}, that is used in the previous search attempt, the time of the previous search attempt (T_{prv}), and the number of discovered paths in the previous search attempt, NPS_{prv}.



IP Address	SAS_{prv}	T_{prv}	NPS_{prv}

Figure 3.8: Previous Searching Area Size (PSS) Table

3.5.2 The Packet Types and Formats

Packets are used to send and receive information between nodes in the network. *ATL-QoS* framework makes use of seven types of packets. They are the Route REQuest ($RREQ$), Route REPly ($RREP$), Route ERRor ($RERR$), $HELLO$, Acknowledgments (ACK), Destination FeedBack (DFB), and data packets. The packet formats used in the *ATL-QoS* framework are standard ones; *ATL-QoS* adds some additional fields to serve *ATL-QoS* functionality. The remaining part of this section describes the role and formats of these packets types.

3.5.2.1 Route Request Packets (RREQ)

A $RREQ$ packet is sent by a source node, N_S, to initiate a path discovery phase. In doing so, N_S sends the $RREQ$ packet to its neighbouring nodes that forward the packet until it reaches the destination node, N_D.

As shown in Figure 3.9, a $RREQ$ packet contains the following fields, the packet sequence number, the source and destination nodes' unique IP addresses and location information (described in Section 3.5.1.1), the packet sending time, the path list that is described in Section 3.5.1.1, and the Searching Area Size, (SAS_{exs}).

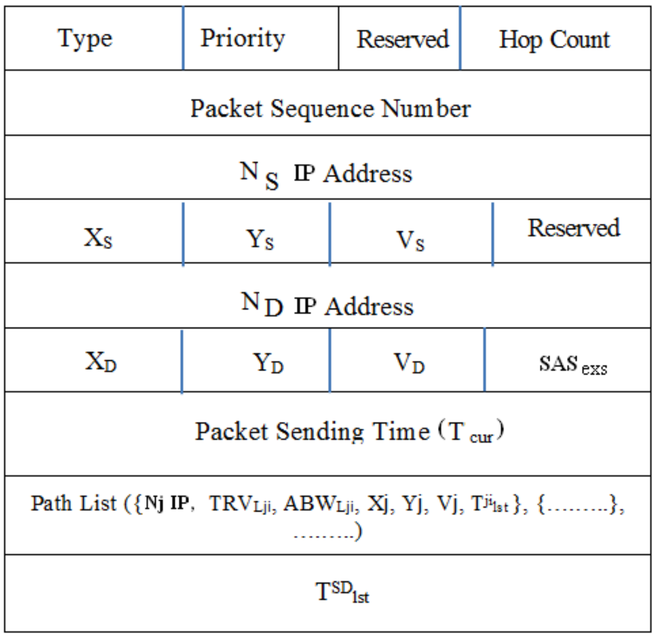
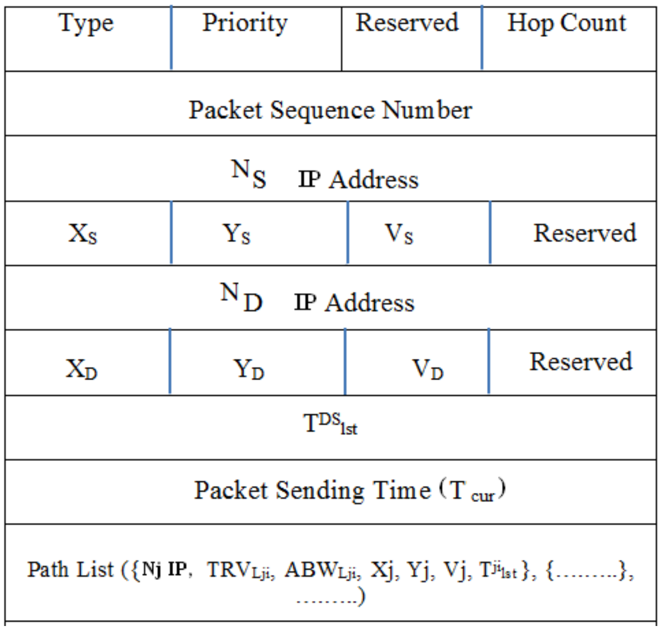


Figure 3.9: *RREQ* Packet Format

3.5.2.2 Route Reply Packets (RREP)

A *RREP* packet is sent by the destination node, N_D, during the path discovery phase. As shown in Figure 3.10, the *RREP* packet contains the packet sequence number, the source and destination nodes' unique IP addresses and location information, the received *RREQ* packet sequence number, the packet sending time, and the path list which contains the information of all the intermediate nodes forming the path from the source node to the destination node.



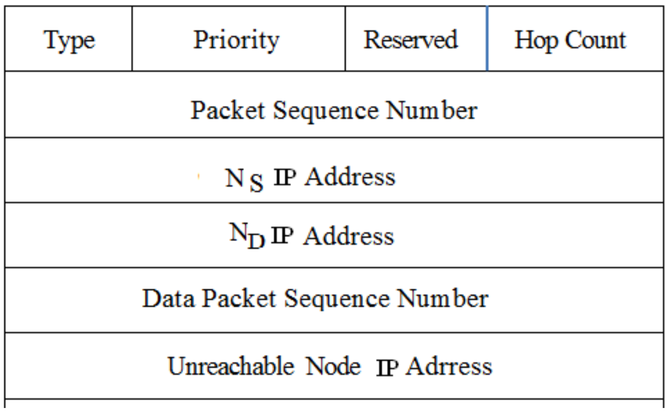
Type	Priority	Reserved	Hop Count
Packet Sequence Number			
N_S IP Address			
X_S	Y_S	V_S	Reserved
N_D IP Address			
X_D	Y_D	V_D	Reserved
T^{DS}_{1st}			
Packet Sending Time (T_{cur})			
Path List ({Nj IP, TRV_{Lji}, ABW_{Lji}, Xj, Yj, Vj, T^{ji}_{1st}}, {.........}, ...)			

Figure 3.10: *RREP* Packet Format

3.5.2.3 Route Error Packets (RERR)

A *RERR* packet is sent by an intermediate node to report a broken link along the path from the source to the destination node during the data transmission phase. During a data transmission phase, when an error occurs at any intermediate node, N_j, along the path, N_j will send a *RERR* packet to N_S, informing N_S that N_j has detected an error while attempting to transmit the traffic to N_i.

As shown in Figure 3.11, the *RERR* packet contains the following fields: the packet sequence number, the source and destination nodes' unique IP addresses, the data packet sequence number that the node was trying to send, and the IP address of the unreachable node.



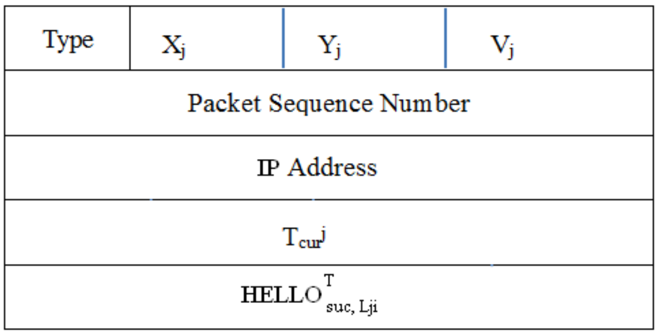
Type	Priority	Reserved	Hop Count
Packet Sequence Number			
N_S IP Address			
N_D IP Address			
Data Packet Sequence Number			
Unreachable Node IP Adrress			

Figure 3.11: *RRER* Packet Format

3.5.2.4 HELLO Packets

A $HELLO$ packet is used by each node to discover its neighbouring nodes. Each node periodically broadcasts $HELLO$ packets to its one-hop neighbouring nodes.

As shown in Figure 3.12, a $HELLO$ packet contains the packet sequence number, this node's unique IP address and its location information. It also contains the $HELLO^T_{suc,Lji}$ that is going to be used to estimate the link available bandwidth, as explained in Section 4.4.



Type	X_j	Y_j	V_j
Packet Sequence Number			
IP Address			
T_{cur}^j			
$HELLO^T_{suc, Lji}$			

Figure 3.12: *HELLO* Packet Format

3.5.2.5 Acknowledgments Packets (ACK)

An ACK packet is a one-hop packet that is sent, by each node, to acknowledge the receipt of control (except the $HELLO$ packets) and data packets. A node, upon the receipt of a data or control packet, will send back an ACK packet to the sender of the packet (one-hop neighbour) to confirm that the packet has reached the next hop.

As shown in Figure 3.13, an ACK packet contains the following fields, a packet sequence number, the receiver node IP address, the sender node IP address and the original packet sequence number. The sender node IP address and the original packet sequence number are used to identify the packet acknowledged by this ACK packet.

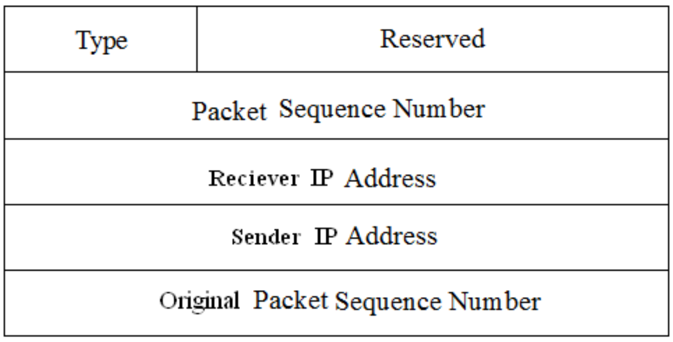
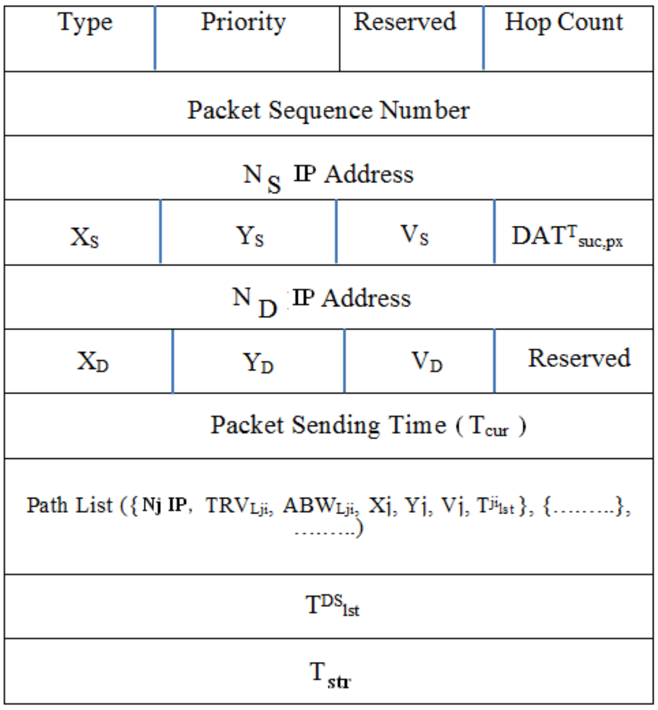


Figure 3.13: ACK Packet Format

3.5.2.6 Destination FeedBack Packets (DFB)

A DFB packet is sent by the destination node to the source node to acknowledge the reception of the data packets. It is an end-to-end data acknowledgment sent during the data transmission phase. $ATL\text{-}QoS$ uses the DFB packet to validate and adjust the path's available bandwidth and trust values discovered during path discovery phase and then reallocate the data traffic based on the new values.

As shown in Figure 3.14, the DFB packet contains the following fields, a packet sequence number, the source and destination nodes' unique IP addresses and location information, the packet sending time, the path list of all the intermediate nodes forming the path, $DAT^{T}_{suc,Px}$ that is the number of the data packets that has been received by the destination node via P_x during the period T, and the starting counting time, T_{str}.

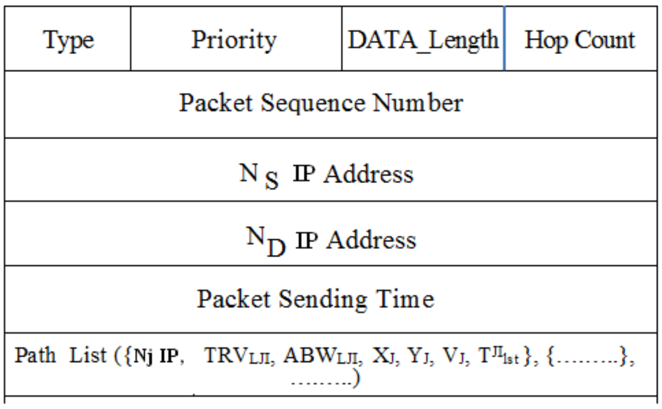


Type	Priority	Reserved	Hop Count
Packet Sequence Number			
N_S IP Address			
X_S	Y_S	V_S	$DAT^T_{suc,px}$
N_D IP Address			
X_D	Y_D	V_D	Reserved
Packet Sending Time (T_{cur})			
Path List ({ Nj IP, TRV_{Lji}, ABW_{Lji}, Xj, Yj, Vj, T^{ji}_{lst} }, {………}, ………)			
T^{DS}_{lst}			
T_{str}			

Figure 3.14: *DFB* Packet Format

3.5.2.7 Data Packets

As shown in Figure 3.15, the fields of a data packet are the packet sequence number, the source and destination nodes IP addresses, the packet sending time, that is used here in calculating the packet end-to-end delivery delay, and the path list of all the intermediate nodes forming the path.



Type	Priority	DATA_Length	Hop Count
Packet Sequence Number			
N_S IP Address			
N_D IP Address			
Packet Sending Time			
Path List ({Nj IP, TRV_{LJI}, ABW_{LJI}, X_J, Y_J, V_J, T^{JI}_{lst} }, {………}, ………)			

Figure 3.15: *Data* Packet Format

3.5.3 Local Information Management

This component maintains the attribute values in the routing table and NNI table. It consists of the following subcomponents:

- Network Nodes Location Information Dissemination Method.
- Neighbouring Nodes Discovery Method.
- Available Bandwidth Estimation Method.
- Trust Value Estimation Method.

The interactions among these subcomponents are shown in Figure 3.16. The subcomponents are explained in the following subsections.

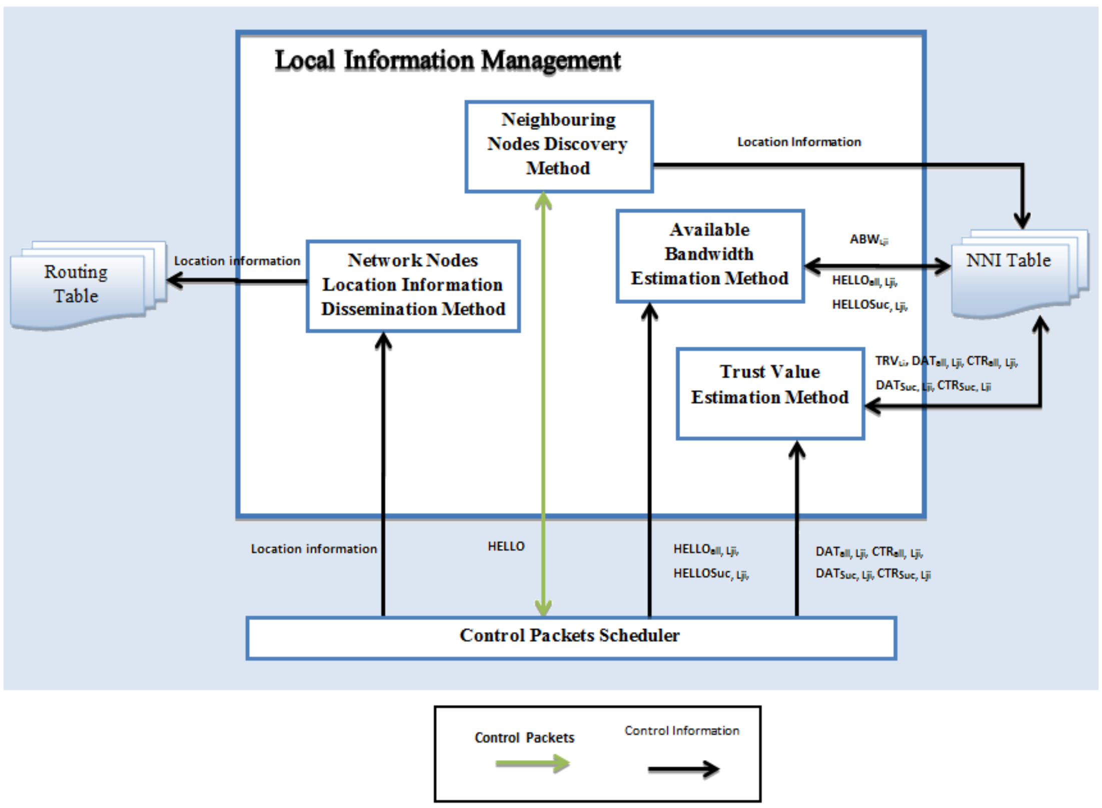

Local Information Management
Neighbouring Nodes Discovery Method
Location Information
Routing Table
Location information
Network Nodes Location Information Dissemination Method
Available Bandwidth Estimation Method
ABW_{Lji}
$HELLO_{all}, L_{ji},$
$HELLOSuc, L_{ji},$
NNI Table
Trust Value Estimation Method
$TRV_{Li}, DAT_{all}, L_{ji}, CTR_{all}, L_{ji},$
$DAT_{Suc}, L_{ji}, CTR_{Suc}, L_{ji}$
Location information
HELLO
$HELLO_{all}, L_{ji},$
$HELLOSuc, L_{ji},$
$DAT_{all}, L_{ji}, CTR_{all}, L_{ji},$
$DAT_{Suc}, L_{ji}, CTR_{Suc}, L_{ji}$
Control Packets Scheduler
Control Packets
Control Information

Figure 3.16: Local Information Management Components

3.5.3.1 Network Nodes Location Information Dissemination Method

This method maintains the network node location information attribute values, as explained in Section 3.5.1.1, in the routing table. Each node should know its physical location by using methods such as the Global Position System (GPS). Each node disseminates its location information to its neighbours by periodically broadcasting $HELLO$ packets containing its location information to its one-hop neighbours. Upon receiving the $HELLO$ packets, the receiving node adds or updates its NNI table with the received information.

In addition, each node appends its location information to any packet they send (i.e. $RREQ$, $RERR$, $RREP$, ACK, DFB, data packets) or re-send (i.e. $RREQ$ packets). Upon receiving such packets, the receiving node uses the location information value, carried in that packet, to add/update the corresponding location information attribute values in its routing table.

3.5.3.2 Neighbouring Nodes Discovery Method

Neighbouring nodes are discovered and maintained by using $HELLO$ packets. To discover neighbours, each node periodically broadcast a $HELLO$ packet to its neighbours. Upon receiving the $HELLO$ packet, a receiving node adds or updates its NNI table with the received neighbour information.

If a node does not receive a $HELLO$ packet from a particular neighbour N_i within a specified time period, the node will consider N_i as unreachable and delete its record from the NNI table.

3.5.3.3 Available Bandwidth Estimation (ABE) Method

This method estimates the available bandwidth of a link between any node N_j, and its neighbour node N_i, ($ABW_{L_{ji}}$). This is done periodically by using the method explained in Section 4.4.

3.5.3.4 Trust Value Estimation (TVE) Method

The TVE method estimates a trust value for each neighbouring node by using a local trust model. This is done periodically by using the method explained in Section 4.3.

3.5.4 Paths Discovery

The Path Discovery component discovers multiple path from the source node, N_S, to the destination node, N_D. It uses the novel protocol, $TADLV2$, that discovers the most reliable set of multiple path between a pair of nodes that reduces the cost in the presence of node mobility and packet dropping attacks. The $TADLV2$ protocol, described in Chapter 6, is an improved version of our early designed Protocol, $TADL$, described in Chapter 5.

3.5.5 Path Selection and Traffic Allocation

The Path Selection and Traffic Allocation component selects a path or a set of the most reliable path(s) from the multiple path that have been discovered during the previous path discovery phase and allocates the traffic to the selected paths. The component uses a novel solution, SAV which is explained in Chapter 7, to accomplish this task. In other words, the source node selects a path or a set of paths from the discovered paths and allocates traffic accordingly based on the required bandwidth by the application and the trust values and bandwidth capacities of the selected paths.

3.5.6 Path Quality Verification

The Path Quality Verification component is responsible for adapting the *ATL-QoS* framework to a dynamic MANET environment. It uses the SAV solution, explained in Chapter 7. SAV uses feedback packets from the destination node that sends during the data transmission phase. The Path Discovery component is used by the source and destination node functions and operates only during the data transmission phase.

3.5.7 Control Packet Scheduler

The Control Packet Scheduler component verifies the packets that are received and sent from/to the node.

3.5.8 Packets Dispatcher

The Packet Dispatcher component performs the following functions:

- It constructs outgoing packets, data, $HELLO$, $RERR$, $RREQ$, $RREP$ and DFB for transmission, i.e. it creates packet headers for these packets and attaches them to the respective payload. In creating the header, it obtains some attribute values from the control plane and added to the header.

- It receives incoming packets and generates the *ACK* packets to acknowledge their reception.
- In an event when a link break is detected during a data transmission phase, it generates an *RERR* packet to notify the source node of the breakage.
- It forwards the generated outgoing as well as the received incoming packets to the Forwarding Engine.

3.5.9 Forwarding Engine

The Forwarding Engine component performs two functions. The first is to determine how and where a packet should be forwarded. This is achieved by reading the routing decision from the control plane. The decision also contains the full path of intermediate nodes IP addresses which the packet should be forwarded. Depending on the routing decision, the data packet may be duplicated and assigned to more than one path before being queued.

The second function is to process incoming packets when the node is the destination of the received packet. The component will extract the packet header and pass them to the Control Packet Scheduler. If the received packet is a data packet, the component will send it to the application layer.

3.5.10 Output Queue

The Output Queue performs buffer management and scheduling of all the packets to be forwarded by the node. This buffer is used to regulate the forwarding of the packets before they are passed down to the data link layer.

3.6 ATL-QoS Framework Operations

This section describes the operations of the *ATL-QoS* framework. The operations are in two phases: path discovery phase and data transmission phase. This section describes these phases and how the functional components described in Section 3.5 work collectively to provide the *ATL-QoS* framework function.

3.6.1 Path Discovery Phase

The operations in the path discovery phase are for discovering multiple path with minimum communication overheads, selecting one or more paths among the discovered paths and allocating the traffic over the selected paths. The following describes the path discovery phase operations respectively performed by source, intermediate and destination nodes.

3.6.1.1 Source Node Operations

As shown in Algorithm 3.1, when a source node, N_S, has traffic to send to a destination node, N_D, N_S searches the routing table to see if there are already paths to the N_D. If yes, N_S uses these paths to transmit the traffic. If not, a path discovery process is initiated, i.e. the novel protocol, $TADLV2$, explained in Chapter 6, is used. $TADLV2$ is designed to discover the most reliable paths between a pair of source and destination nodes with minimum overhead costs. Once multiple path are discovered, N_S invokes the novel solution, SAV, explained in Chapter 7. SAV selects one path or a set of node-disjoint paths from the discovered paths that satisfy the required bandwidth. Once the paths are selected, N_S allocates the data traffic over the selected path(s). The traffic is allocated to each path based on the quality of the path (i.e. its available bandwidth and trust value). Then, N_S starts the data transmission phase in which the data packets are sent to the destination node along the selected path(s). Alternatively, in SAV, if no path could be found and selected or no paths could collectively satisfy the required bandwidth, a new path discovery phase will be initiated.

Algorithm 3.1 The *ATL-QoS* Source Node Operations During Path Discovery Phase

```
procedure ATL-QoS-S-Discovery($IPaddress, BW_{req}, Priority, data$)
    $SAS_{exs}$= Null
    Search routing table for paths to $N_D$
    if Paths to $N_D$ are found then
        $DiscoveredPaths \leftarrow pathstoN_D$
    else
        $DiscoveredPaths \leftarrow$ TADLV2-S($IPaddress, SAS_{exs}, BW_{req}$)
    end if
    SAV-S-Discovery($BW_{req}, data, DiscoveredPaths, IPaddress, SAS_{exs}$)
end procedure
```

3.6.1.2 Intermediate Node Operations

During the path discovery phase, intermediate nodes are responsible for forwarding *RREQ* packets towards the destination node until the destination node is reached. They do not reply to the *RREQ* packets back to the source node. The intermediate nodes are also responsible for preventing *RREQ* looping. As shown in Algorithm 3.2, upon receiving a *RREQ* packet, the intermediate node executes the novel solution $TADLV2$ (Chapter 6). $TADLV2$ switches between the two node-disjoint path discovery algorithms, the Source node Controlled Multiple Node-disjoint Path Discovery (*SC-MNPD*) algorithms and the Intermediate node Controlled Multiple Node-disjoint Path Discovery (*IC-MNPD*) algorithms, based on the network nodes' mobility level. Upon accepting the *RREQ* packet, the intermediate node adds the values of its attributes in the *RREQ* header. It then forwards the *RREQ* packet on to the chosen downstream neighbours until the destination node is reached.

Algorithm 3.2 The *ATL-QoS* Intermediate Node Operations During Path Discovery Phase

```
procedure ATL-QoS-I-Discovery(RREQ)
    TADLV2-I(RREQ)
end procedure
```

3.6.1.3 Destination Node Operations

During a path discovery phase, the destination node may receive *RREQ* packets. As shown in Algorithm 3.3, when the destination node receives a *RREQ* packet, it invokes the $TADL$ protocol, explained in Chapter 5. In $TADL$, N_D constructs a reply, *RREP* packets and then sends the *RREP* packets to the source node via the reverse of the path.

Algorithm 3.3 The *ATL-QoS* Destination Node Operations During Path Discovery Phase

```
procedure ATL-QoS-D-Discovery(RREQ)
    TADL-D(RREQ)
end procedure
```

3.6.2 Data Transmission Phase

The operations in the data transmission phase cover transmitting the data over the selected set of paths. This section explains the operations carried out by source, intermediate and destination nodes during the data transmission phase.

3.6.2.1 Source Node Operations

During the data transmission phase, the source node is responsible for verifying the paths and maintaining the paths. In this phase, the source node may receive two types of packet, $RERR$ packets or DFB packets. As shown in Algorithm 3.4, when a $RERR$ packet is received, it indicates that an intermediate node detected a link failure during this data transmission phase. In response, the source node deletes the path containing the link in error indicated by the $RERR$ packet from its routing table. It also deletes all the paths to any destination node that contains the failed link. A path discovery phase is then re-performed if the aggregated available bandwidth of the set of the remaining paths, available for data transmission in this data transmission phase, is less than the required bandwidth.

Algorithm 3.4 The *ATL-QoS* Source Node Operations During Data Transmission Phase

```
procedure ATL-QoS-S-Transmission($IP, BW_{req}, Priority, data, RRER$)
    if $RRER$ packet is received then
        $N_S$ deletes all the paths to any destination node that contains that link
from its routing table
        if $ABW_S$ of the used set $<BW_{req}$ then
            ATL-QoS-S-Discovery($IPaddress, BW_{req}, Priority, data$)
        end if
    else
        $Flag \leftarrow$ SAV-S-Transmission($data$)
        if $Flag ==$ False then
            ATL-QoS-S-Discovery($IPaddress, BW_{req}, Priority, data$)
        end if
    end if
end procedure
```

When the source node receives a DFB packet, it invokes the SAV solution, explained in Chapter 7. The SAV solution compares the number of data packets received by the destination node, that is carried in the DFB packets, with the number of data packets sent by the source node during T time period. It uses this comparison result to adjust the path(s) available bandwidths and trust values, and adjusts the number of data packets assigned to each of the selected paths.

3.6.2.2 Intermediate Node Operations

During the data transmission phase, intermediate nodes are responsible for forwarding the data packets towards the destination node. When a link break occurs during the data transmission phase, the intermediate node which discovers the failure will send a *RERR* packet, reporting the link failure to the source node. The intermediate nodes are also responsible for forwarding the DFB packets towards the source node.

3.6.2.3 Destination Node Operations

During a data transmission phase, the destination node receives data packets. As shown in Algorithm 3.5, when the destination node receives the data packets, during the data transmission phase, it invokes the SAV solution, explained in Chapter 7. In SAV solution, the destination node is responsible for counting the number of data packets received via each path and sends a feedback packet containing the counted number of received data packets to the source node, as explained in Chapter 7.

Algorithm 3.5 The *ATL-QoS* Destination Node Operations During Data Transmission Phase

```
procedure ATL-QoS-D-Transmission($data$)
    SAV-D-Transmission($data$)
end procedure
```

3.7 Chapter Summary

This chapter has given an overview of the framework design to support *QoS* requirements in the presence of data packet dropping attack. The *ATL-QoS* framework tries to overcome some weaknesses exhibited by existing routing algorithms used in ad hoc networks. The *ATL-QoS* framework design uses two novel protocols. *ATL-QoS* uses the novel $TADLV2$ protocol that is designed to reduce the routing overhead while discovering trustworthy paths. In addition, the framework uses the novel SAV solution, that uses path selection criteria to select the best possible set of paths that could best satisfy a given *QoS* requirement in adaptation to the network condition and allocate the data traffic on the selected paths. *ATL-QoS* also uses the novel SAV solution to validate and adjust the path's quality estimated during path discovery phase and then reallocate the data traffic based on the new values.

The following chapters present and evaluate the novel protocols used in *ATL-QoS* framework design in detail.

Chapter 4

Building Blocks and Evaluation Methodology

4.1 Chapter Introduction

This chapter describes the building blocks used in the design of the *ATL-QoS* framework. These building blocks are the Trust Value Estimation (TVE) method that is used to estimate the trust value of a link between two nodes, and the Available Bandwidth Estimation (ABE) method that is used to estimate the available bandwidth of a link between two nodes. It also justifies the use of simulation method to evaluate the framework.

In detail, Section 4.2 describes the notations used in the description of the building blocks. Section 4.3 describes the TVE method. Section 4.4 describes the ABE method. Section 4.5 discusses potential evaluation methods thus justifying the use of simulation as the evaluation method. Section 4.6 describes the simulation model used in our framework. Section 4.7 describes how the evaluation model is validated. Section 4.8 describes how statistically significant results are obtained. Finally, section 4.9 concludes the chapter.

4.2 Notations

This section gives the notations used in the building blocks descriptions and then used throughout the thesis. The notations are summarised in Table 4.1.

Table 4.1: Notations Used in the Building Blocks Descriptions

Notation	Definitions
T_0	Initial instance of time
T_{lst}^{SD}	The last time N_S heard from N_D
T_{rcv}	Packet receiving time
R	Radius of a node predicted location
$TRV_{L_{ji}}^{T0}$	Initial trust value for the link L_{ji} linking node j to node i
$TRV_{L_{ji}}^{Tcur}$	The trust value for the link L_{ji} linking node j to node i estimated in the current instant of time, T_{cur}
$TRV_{L_{ji}}^{Tcur-1}$	The trust value for the link L_{ji} linking node j to node i estimated in the previous instant of time, T_{cur-1}
$CTR_{suc,Lji}^{T}$	The number of successfully delivered control packets over the link L_{ji} during the slot T
$CTR_{fal,Lji}^{T}$	The number of failed control packets over the link L_{ji} during the slot T
$CTR_{all,Lji}^{T}$	The number of sent control packets over the link L_{ji}, $CTR_{all,Lji}^{T} = CTR_{suc,Lji}^{T} + CTR_{fal,Lji}^{T}$ during the slot T
$DAT_{suc,Lji}^{T}$	The number of successfully delivered data packets over the link L_{ji} during the slot T
$DAT_{fal,Lji}^{T}$	The number of failed data packets over the link L_{ji} during the slot T
$DAT_{all,Lji}^{T}$	The number of sent data packets over the link L_{ji}, $DAT_{all,Lji}^{T} = DAT_{suc,Lji}^{T} + DAT_{fal,Lji}^{T}$ during the slot T
OBS	OBServation value for packets transmission
OBS_{Lji}^{T}	The OBS value of the packets transmitted over the link L_{ji} during the slot T
$OBS_{Lji,ctr}^{T}$	The OBS value of the control packets transmitted over the link L_{ji} during the slot T
$OBS_{Lji,dat}^{T}$	The OBS value of the data packets transmitted over the link L_{ji} during the slot T
$\triangle TRV$	The increment value of trust
THR_{obs}	Observation value threshold
W_{dat}	Weight assigned to data packets
W_{ctr}	Weight assigned to control packets
ETX_{ji}	A metric value used to estimate packet loss ratio of the link between N_j and its neighbouring node N_i
d_{ji}	Packet delivery ratio for the packet delivered from N_j to N_i
ABW_{Lji}^{T}	The available bandwidth for the link L_{ji} during the slot T
ABW_{Lji}^{Tcur}	The available bandwidth for the link L_{ji} linking node j to node i estimated in the current instant of time, T_{cur}
ABW_{Lji}^{Tcur-1}	The available bandwidth for the link L_{ji} linking node j to node i estimated in the previous instant of time, T_{cur-1}
$HELLO_{suc,Lji}^{T}$	The number of $HELLO$ packets successfully transmitted over the link L_{ji} during the slot T
$HELLO_{all,Lji}^{T}$	The number of sent $HELLO$ packets transmitted over the link L_{ji} during the slot T
DLY	Average End-to-End Packet Delivery Delay
PDR	Packet delivery ratio

4.3 Trust Value Estimation Method

This section presents an overview of the Trust Value Estimation (TVE) method used in the *ATL-QoS* framework. As mentioned earlier, one of the tasks each node ought to perform in *ATL-QoS* is to estimate a trust value for each of its neighbours. This is done by using the local trust estimation method explained in [88]. In this method, each node estimates a trust value for each of its neighbouring nodes based on observing the reliability of the link connecting the node to the neighbouring nodes. At each node N_j, the trust value assigned to a neighbouring node, N_i, reflects the reliability of the link linking N_j and N_i. The higher the trust value a node assigns to a neighbouring node, the more reliable the assigned node is. This trust value is then used for selecting a path(s) with a higher level of reliability. A link trust value is measured in a given range [TRV_{min}, TRV_{max}], where TRV_{min} is the minimum trust value, TRV_{max} is the maximum trust value.

The trust value N_j assigned to N_i at the current instance of time is denoted as $TRV_{L_{ji}}^{Tcur}$. The value of $TRV_{L_{ji}}^{Tcur}$ is estimated as follows. Initially, when N_i newly joins the neighbourhood of N_j, N_j assigns a neutral value to $TRV_{L_{ji}}^{T0}$, which is calculated using Equation 4.1.

$$TRV_{L_{ji}}^{T0} = \frac{TRV_{min} + TRV_{max}}{2} \tag{4.1}$$

where TRV_{min} is the minimum trust value, TRV_{max} is the maximum trust value specified for our framework. The TRV_{max} value is set to 10 and the TRV_{min} value is set to 0 (as recommended in [88]). T_0 is the time when N_i joins the neighbourhood of N_j. This neutral value (calculated using Equation 4.1) indicates that N_j cannot yet determine whether the link connecting to this new neighbour is trustworthy or not. This value is assigned to $TRV_{L_{ji}}^{Tcur}$. Then, from this point on, the value of $TRV_{L_{ji}}^{Tcur}$ is updated by N_j regularly at each interval of T, where T = T_{cur-1} - T_{cur} based on the value of OBS_{Lji}^{T}, where OBS_{Lji}^{T} is the direct observation value for packet transmission that captures the reliability of the link, L_{ji} during the interval of time, T. OBS_{Lji}^{T} is calculated based on the number of acknowledgment packets successfully received from neighbour node, N_i, during the interval T. It is calculated using Equation 4.2.

$$OBS_{Lji}^{T} = (OBS_{Lji,ctr}^{T} \times W_{ctr}) + (OBS_{Lji,dat}^{T} \times W_{dat}) \tag{4.2}$$

where $OBS_{Lji,ctr}^{T}$ and $OBS_{Lji,dat}^{T}$ are, respectively, the direct observation value of the link, L_{ji}, in terms of control and data packets, and W_{ctr} and W_{dat} are the weighting values assigned to $OBS^{T}_{Lji,ctr}$ and $OBS^{T}_{Lji,dat}$, respectively. The sum of these values, i.e. $(W_{ctr} + W_{dat})$, should be 1. The determination of W_{dat} and W_{ctr} values will be explained in detail in Section 4.8.3.

$OBS^{T}_{Lji,ctr}$ is calculated as:

$$OBS^{T}_{Lji,ctr} = \frac{CTR^{T}_{suc,Lji} - CTR^{T}_{fal,Lji}}{CTR^{T}_{all,Lji}} \tag{4.3}$$

where $CTR^{T}_{suc,Lji}$ is the number of control packets that are sent by N_j to N_i during the same interval T and are successfully acknowledged by N_i. The control packets are route request, route reply and route error packets. $CTR^{T}_{fal,Lji}$ is the number of control packets that are sent by N_j to N_i during the same interval but failed to be acknowledged by N_i. $CTR^{T}_{all,Lji}$ is the total number of transmitted control packets by N_j to N_i during the interval T; it is the sum of $CTR^{T}_{suc,Lji}$ and $CTR^{T}_{fal,Lji}$.

Similarly, $OBS^{T}_{Lji,dat}$ is calculated as:

$$OBS^{T}_{Lji,dat} = \frac{DAT^{T}_{suc,Lji} - DAT^{T}_{fal,Lji}}{DAT^{T}_{all,Lji}} \tag{4.4}$$

The OBS^{T}_{Lji} value indicates the reliability of the link (linking N_j to N_i) when delivering the control and data packets from N_j to N_i during the interval, T. If the link is less reliable, i.e. fewer packets are acknowledged, the OBS^{T}_{Lji} value will be lower, indicating that N_i and the link connecting to N_i are less trustworthy. In *ATL-QoS*, depending on this observation value, OBS^{T}_{Lji}, N_j updates the value of TRV^{Tcur}_{Lji} based on Equation 4.5.

$$TRV^{Tcur}_{Lji} = \begin{cases} TRV^{Tcur-1}_{Lji} + \triangle TRV & ,OBS^{T}_{Lji} >= THR_{obs} \\ TRV^{Tcur-1}_{Lji} - \triangle TRV & ,OBS^{T}_{Lji} < THR_{obs} \end{cases} \tag{4.5}$$

where $\triangle TRV$ is an increment value. The $\triangle TRV$ value is set to 0.1 (as obtained experimentally in [88]). THR_{obs} is an observation threshold value. This value is configurable during experiments as described in Section 4.8.2. Looking at the equation, we can see that the trust value continues to increase or decrease unless it reaches the maximum value of 10 or minimum value of 0.

4.4 Available Bandwidth Estimation Method

This section describes a method used in the *ATL-QoS* framework for estimating the available bandwidth between a pair of nodes. The method is called the Available Bandwidth Estimation (*ABE*) method. One of the tasks each node performs in *ATL-QoS* is to estimate the available bandwidth for each of its neighbours. In an ideal situation, i.e. where all the transmitted packets are received, the bandwidth (measured in terms of packets/s) is equal to the traffic transmission rate, $Traffic\_Transmission\_Rate$, which is set to 4 packets/s as specified in Section 4.6. In reality, however, a link may not be able to transmit the traffic with this rate, so we need to estimate the available bandwidth of a link. The available bandwidth often characterises the amount of traffic that the network can transfer per second [91]. To estimate the available bandwidth of a link linking N_i to its neighbouring node N_j, we need to measure the link expected transmission ratio, then estimate the link available bandwidth, ABW^{T}_{Lji} using Equation 4.6.

$$ABW^{T}_{Lji} = Link\_Expected\_Transmission\_Ratio \times Traffic\_Transmission\_Rate \tag{4.6}$$

The most relevant method in the literature for measuring the link expected transmission ratio is the Expected Transmission Count (ETX) routing protocol [22, 51, 84]. The ETX protocol is a routing protocol that is commonly used in MANETs. The protocol is design to find a path that can provide sufficient available bandwidth to transmit traffic through that path. Simulation results have shown that the ETX approach is more effective in terms of finding a better path than the popular minimum hop count approach, particularly for paths with two or more hops [84].

The protocol uses an ETX metric to estimate the expected number of transmissions of the links forming the path. The ETX value assigned to N_i reflects the maximum achievable transmission rate (including retransmissions) of the link linking this node, N_j, and N_i. It is computed using a forward delivery ratio and a reverse delivery ratio of every link along the path. The forward delivery ratio of N_j to N_i, donated as d_{ji}, is the probability that a data packet is successfully delivered to the neighbouring N_i by N_j. The reverse delivery ratio at N_j, donated as d_{ij}, is the probability that N_j successfully receives a packet from the neighbour node N_i. In ETX, these probabilities are calculated by using probe packets.

Nodes exchange their probe packets with their neighbours. They calculate the delivery ratios to find the ETX metric of a link. The calculation of an ETX metric value for a link can be described in three steps.

Step 1: When a node joins the network, it starts broadcasting probe packets to all of its neighbouring nodes and continues to do so periodically. Each probe packet contains the IP address of the node that sends the packet. The rate of broadcasting the probe packets is one packet/s (as specified in ETX protocol [22]). The node does this until it leaves the network.

Step 2: When N_j receives the first probe packet from N_i, N_j starts to count the number of probe packets that N_j receives from N_i within the past SYNC-TIME seconds of the duration. The recommended SYNC-TIME value is 15 s (as described in the ETX [22]). After SYNC-TIME seconds, N_j calculates a reverse delivery ratio, d_{ij}. d_{ij} is calculated as the number of received packets divided by $BROADCAST\_Packets$ (as shown in Equation 4.7). $BROADCAST\_Packets$ represents the number of sent probe packets in SYNC-TIME seconds.

$$reverse\_delivery\_ratio = \frac{received\_packets}{BROADCAST\_Packets} \tag{4.7}$$

Similarly, N_i also counts the received probe packets sent from N_j and calculates a reverse delivery ratio, d_{ji}, using the same equation, Equation 4.7. This reverse delivery ratio of N_i is the forward delivery ratio of N_j. In other words, N_j now has its reverse delivery ratio, d_{ij}, and N_i has N_j's forward delivery ratio, d_{ji}.

Step 3: Nodes exchange their respective reverse delivery ratios with each of their neighbouring nodes. Once N_j receives the forward delivery ratio d_{ji} from N_i, N_j calculates the ETX metric, that is the expected number of transmissions, of the link linking N_j and N_i by using Equation 4.8.

$$ETX_{ji} = \frac{1}{d_{ji} \times d_{ij}} \tag{4.8}$$

In an ideal case, each of the two nodes could receive all the probe packets sent by the other node. In this case, the ETX value of the link linking the two nodes is one as (1 =1/((15/15) × (15/15)). All of the data packets sent through the link are predicted to be delivered successfully. In reality, however, a node may not receive all the probe packets sent by the other node. In this case, the delivery ratios can be under 1 which increases the ETX value. So, a link with a lower

ETX value is preferable when making a routing decision.

Equation 4.8 is used to calculate an ETX value for one link between N_j and N_i. The ETX value of a path is the sum of the ETX values of all the links forming the path. This means that a path with more hops will be likely to have a higher ETX value. Therefore, in a path selection process, a path with a lower ETX value is more likely to have fewer hops, thus, more preferable.

The ETX protocol has two limitations if used in our framework. Firstly, it assumes that the quality of a link is symmetric in terms of delivery ratios, i.e. the delivery ratios for both directions of the link are assumed to be the same. This assumption is not always true for wireless links. Real-world wireless links are unidirectional.

Secondly, the ETX protocol takes the hop count into consideration when it sums up the ETX values of the links forming the path, and a path with a lowest ETX value will be selected. With this protocol, a path with a higher number of hops but formed by most trustworthy links, may not be selected even if it is of a better quality than a path with a fewer number of hops but formed by lossy links.

These two limitations make this method not directly applicable to the *ATL-QoS* framework. We have designed a new method, called Available Bandwidth Estimation (*ABE*) method inspired by the ETX method. The ABE method has three major differences from the ETX method. Firstly, ABE does not use separate probe packets, rather it makes use of the $HELLO$ packets already used. In other words, in ETX, the forward delivery ratio values are calculated based on how probe packets are sent and received, but in ABE, the forward delivery ratio values are calculated based on the $HELLO$ packets. This is because $HELLO$ packets are already being used for neighbouring node discovery, so using these packets to estimate ETX metric values can prevent introducing extra control overheads into the underlying network. As mentioned earlier, in *ATL-QoS*, each node periodically broadcasts $HELLO$ packets to its neighbours containing its IP address and location information. In this modification, when a node N_i receives the first $HELLO$ packet from N_j, N_i will start counting the number of $HELLO$ packets received from N_j during that duration T, and record this count in a $HELLO^T_{suc,Lji}$ fields in the NNI table. The value of T, here, is set to SYNC-TIME seconds (as described in the ETX [22]).

The second modification is that, in ABE, we do not assume that a link is symmetric. In other words, a delivery ratio calculated in one direction of a link is only applicable to the link in one direction, e.g. from N_j to N_i. In this modification, upon the expiry of SYNC-TIME, N_i sends the $HELLO^T_{suc,Lji}$ in a $HELLO$ packet to N_j. The use of the $HELLO$ packet to carry the $HELLO^T_{suc,Lji}$ can prevent introducing extra control overheads into the underlying network. In addition, each node will calculate its own forward delivery ratio, named as the link expected transmission ratio, using the $HELLO^T_{suc,Lji}$ for the link from N_j to N_i. When N_j receives $HELLO^T_{suc,Lji}$, it calculates the link expected transmission ratio using Equation 4.9.

$$Link\_Expected\_Transmission\_Ratio = \frac{HELLO^T_{suc,Lji}}{BROADCAST\_Per\_T} \tag{4.9}$$

where $HELLO^T_{suc,Lji}$ is the number of $HELLO$ packets received by N_i sent by N_j, $BROADCAST\_Per\_T$ represents the number of $HELLO$ packets broadcasted by N_j during T. Then, after calculating the link expected transmission ratio, the link available bandwidth, ABW^T_{Lji} is estimated. ABW^T_{Lji}, which is the available bandwidth that N_j assigns to N_i for period T, is estimated using Equation 4.6. If the link is in an ideal state, the two nodes will receive all the $HELLO$ packets sent by the other node. In this case, the link expected transmission ratio value of the link will be 1 (as 1 =15/15). However, in reality, for example, node N_j may not receive all the $HELLO$ packets sent by N_i. The link expected transmission ratio, in this case, will be less than 1. As a result, the ABW^T_{Lji} value of the link for this direction in the duration T will be lower than this value. In other words, a link with a higher ABW value is preferable when making a routing decision.

The third modification made in ABE is to overcome the second drawback of the ETX protocol. ABE uses the minimum bandwidth of all the links forming the path to be the available bandwidth of the path, $ABW^T_{P_i} = \min^h_{i=1} \left\{ABW^T_{Lji}\right\}$, instead of using the sum of the ETX values as a measure of the path quality. In this way, we do not have the problem caused by considering the hop count. In other words, in ABE, a path with the highest ABW^T_{Px} value, regardless of the hop count, will be chosen when making a routing decision.

4.5 Evaluation Methodology

This section discusses the possible evaluation methods that may be used to evaluate the *ATL-QoS* framework, and this discussion serves as our justification for the selection of the methodology used in the study presented in this thesis. According to literature [82], there are three evaluation methodologies, real system experiments, mathematical modelling and simulation.

4.5.1 Real System Experiments

The real system experiment evaluation method evaluates the performance of any solution designed on a full-scale physical network. It aims at capturing the full interaction between all parts. It uses test-beds that are built over a large physical space because of the wide coverage area of radio signals [3]. The main advantage of this methodology is that it generates experimental data that are based on realistic conditions. This is useful in understanding the behaviour and performance of a protocol or a system before deploying it for general use in large-scale networks [89].

However, this methodology has two drawbacks. Firstly, it is expensive to set up and to operate, as it requires us to set up a network test-bed of physical nodes, to install the software implementing the *ATL-QoS* framework and to collect experimental data using this test-bed. This is not feasible within the time frame of a PhD project. Secondly, the real-world experimental conditions are difficult to control. For example, changes of the outdoor environment are unpredictable and may affect the results collected [89]. Thus, real system experimental evaluation is not used in this thesis.

4.5.2 Mathematical Modelling

The second evaluation method is through the use of a mathematical model of a network and any solution designed for the network. This method is cheaper than the method discussed above, but may not generate valid results. This is because *ATL-QoS* framework is designed for a rather complex ad-hoc network operating in a dynamic environment. To model complex operations, many assumptions have to be made to keep the analysis traceable, and these simplifying assumptions may limit the usefulness or validity of the findings. Thus, mathematical modelling evaluation method is not used in this thesis.

4.5.3 Simulation

The simulation evaluation method uses a network simulator to model the behaviour of a network and any solution designed for the network. For a MANET, this method is attractive and most commonly used. It has two main benefits [50, 87]. Firstly, simulation is scalable, i.e. it can easily be used to simulate a small or a large network. Secondly, the network conditions can be easily controlled. For these reasons the simulation evaluation method is chosen as the evaluation methodology for the work reported in this thesis.

The Network Simulator, *NS-2*, (Network Simulator version 2), has been chosen. *NS-2* is the most widely used network simulator for MANET research [106, 101, 53]. *NS-2* is a discrete event simulator specifically developed for networking research. It has found large acceptance, as a tool, to experiment new ideas, protocols and distributed algorithms. It is open-source are available in the public domain.

NS-2 uses two languages C++ and OTCL. C++ is used for the basic script (protocols and routing algorithms). C++, as a fast programming language, is used to increase the calculating power. OTCL is used to describe simulation conditions (network topology, physical links, protocols being used, traffic generated by the sources, etc.) in the form of scripts. OTCL is used to make the simulator easy to use (e.g. for editing or modifying simulations).

NS-2 has a large number routing algorithms installed, including DSR, $DSDV$, and $AODV$. However, the LAR protocol, which we used for the evaluation of our framework, is not included in the *NS-2* package. The available version of the LAR protocol is published by [55]. It is only available for Ns-2.27.2. For this reason, we have chosen the network simulator NS-2.27.2 [106] as our evaluation tool.

4.6 Simulation Configuration

This section explains simulation configuration, network modelling, performance metrics used in the evaluation of our *ATL-QoS* framework, and explains how some of the parameter values are chosen during our simulation study.

4.6.1 Network Configuration

The network consists of 50 nodes. The nominal transmission range of a node is 250 metres. The wireless standard used is IEEE 802.11 with a data rate of 2Mbps. Nodes are located in a flat 1000m $\times$ 1000m area. A square topology is used as it does not favour one range of motion over any other [40] (unlike, say, a rectangular topology). The duration of each simulation run is 900 s. Using these values are common in simulation studies [19, 45]. Results are averaged over 30 simulation runs with a 2.86 confidence level. Results are averaged to reduce the effects of randomness in the simulation results; this number of simulation runs is determined in Section 4.8.1.

The packet/bit error rate is set to 0, as the focus of this research is packet loss due to attackers, node mobility, and congestion at the network layer, i.e. if non-zero error rate packets is used, corruption and loss due to the lossy wireless environment can occur and this will increase the difficulty of observing the packet loss caused by the factors of concern.

4.6.2 Mobility Model

In our simulation, we have chosen to use the random waypoint mobility model [18]. In this model, each node remains stationary for a pause time, and then randomly selects a location within the network coverage area and moves to this location at a random speed between a minimum and a maximum mobility speed values. After arriving at the location, the node again remains stationary for a pause time before moving to the next randomly chosen location. The minimum mobility speed used is 1m/s (3.6kph), and the maximum speed used is 19m/s (68.4kph). These mobility values are used in the original research using the random waypoint mobility model [19] and this mobility model along with the speed values are also commonly used in the relevant studies published in literature [75]. The pause times used are 0, 300, 600, and 900 s. Setting the pause time to 0 s means the nodes move continuously, whereas for stationary nodes the pause time is set to 900 s. The duration of each simulation run is 900 s, so 900 s is, effectively, a representation of infinity. This process is repeated for the duration of the simulation.

4.6.3 Traffic Patterns

Each node in the network transmits Constant Bit Rate (CBR) at a rate of 4 packets/s. The packet size is 512 bytes. For this given packet size, a four hop count can be supported and this is sufficient for the network size under investigation. However, if a larger network to be investigated, a larger hop count maybe necessary and a larger packet size should be used. The packet format is described in Section 3.5.2. The traffic loads used is generated by 20%, 40%, 60%, 80%, and 100% of the network nodes. These nodes are called the source nodes. 30% of the source nodes generate high priority traffic and the remaining 70% of source nodes generate low priority traffic. These percentage values are given based on the research finding reported in literature [75].

4.6.4 Packet Dropping Attacks

In our simulation model, two types of misbehaviour are captured. The first one is packet dropping attacks which are performed by packet dropping attackers. The second is selfish behaviour that is exhibited by selfish nodes. The packet dropping attackers are malicious nodes. Their intention is to drop data packets. To do so, they must be included in a path, and to be included in a path they must take part in a path discovery phase by forwarding control packets. In other words, this class of attackers forwards control packets and only drops data packets. These attackers do not discriminate between priority and non-priority data packets. They may use different strategies to drop packets [34]; they may choose to discard all the incoming data packets (in which case, the attack is called a black hole attack [93, 116, 81]), or drop them selectively (called a grey hole attack [93]). Selfish nodes, on the other hand, usually lose both data and control packets. The selfish node behaviour is similar to the behaviour exhibited by a buffer overflowing node. Buffer overflowing nodes will be considered as overloaded nodes.

In our simulation model, we use TCL commands to indicate which node is a packet dropping attacker and which node is a selfish node [28]. These TCL commands are shown in the following:

```
$ns_ at 0.1 "$g(0) setDataDrop"
$ns_ at 0.1 "$g(1) setSelfish "
```

This indicates that mobile node 0 will start to act as a packet dropping attacker at time 0.1 s and node 1 will begin to act as a selfish node at time 0.1 s.

The main program of the routing protocol is processed in the method command() as follows:

```
if(strcmp(argv[1], "setDataDrop"))
    node_type ='d'
else
    if(strcmp(argv[1], "setSelfish"))
        node_type ='s'
    else
        node_type ='n'
```

In the above code, the variable node_type indicates the type of behaviour a node performs. If node_type ='d', the node is set to be a packet dropping attacker, if node_type ='s', the node is set to be a selfish node. A normal node will have node_type ='n'. As mentioned above, when a selfish node receives a data or a control packet, it simply drops it. When a packet dropping attacker receives a data packet it selectively drops it, but if it receives a control packet it will process it normally. The recv (Packet *p, Handler *) method is invoked when an intermediate node received a data packet. The following code is added to the method to implement these threat behaviours.

```
dropflag = (rand()RAND_MAX);
if (((node_type =='d')&& (dropflag>=0.5)) || (node_type =='s'))
    drop(p);
```

where dropflag is a random value between 0 and 1. If the variable node_type is set to 'n', which indicates a normal node, the data packets received will be processed and be forwarded. However, for control packet (*RREQ* or *RREP*), the following code is added in the method:

```
if (node_type =='s')
    drop(p);
```

This is how, if the variable node_type is 'n' or 'd', which indicates a normal or a packet dropping node, the control packets received will be processed and forwarded. The control packets are only dropped by selfish nodes (when node_type is 's').

During our simulation study, to be repeated in later chapters, the attacker ratios (i.e. the percentage of nodes that perform the mentioned attacks) are, respectively, set to 0% (representing a non-malicious network environment), 5%, 10%, 15% and 30%.

4.6.5 Performance Metrics

Three performance metrics are used in our investigation of the *ATL-QoS* framework. They are routing overhead, packet delivery ratio, and average end-to-end packet delivery delay.

Routing Overhead [58] refers to the total number of control packets transmitted by all the nodes in the network, CTR_{all}, divided by the total number of data packets successfully received by all the destination nodes, DAT_{suc}. The control packets counted here are Route REQuest ($RREQ$), Route REPly ($RREP$), Route ERRor ($RERR$), Acknowledgments (ACK), and Destination FeedBack (DFB) Packets.

$$Routing\_Overhead = \frac{CTR_{all}}{DAT_{suc}} \tag{4.10}$$

Packet Delivery Ratio (PDR) is the ratio of the total number of data packets successfully received by all the destination nodes, DAT_{suc}, to the total number of data packets transmitted by all the source nodes, DAT_{all}, in the network, i.e.

$$PDR = \frac{DAT_{suc}}{DAT_{all}} \times 100 \tag{4.11}$$

Average End-to-End Packet Delivery Delay (DLY) is the average time difference between when a data packet is transmitted by a source node, (T_{trn}), and when the data packet arrives at its intended destination node, (T_{rcv}), i.e.

$$DLY = T_{rcv} - T_{trn} \tag{4.12}$$

4.7 Simulation Validation

In this section, the simulation model is validated to confirm that the implementation of the *ATL-QoS* framework is correct. It is necessary to validate the simulation model, so that the results, obtained from it, are reliable.

The simulation model built for the *ATL-QoS* framework is validated using six scenarios. The first four scenarios assume that there is no malicious node in the network, and all the nodes are stationary, i.e. the pause time used is 900 s. As there is no malicious behaviour and there is no overflow condition building up in the network, the anticipated PDR value in the four scenarios is 100%. The last two scenarios assume that there is a malicious node in the network. These scenarios are special cases of a network setting. During validation, the network is respectively, set to each of the scenario situation. The results, in terms of packet delivery ratios, routing overheads, and average end-to-end packet delivery delays are examined and compared with anticipated results. The scenarios use the topology in Figure 4.1.

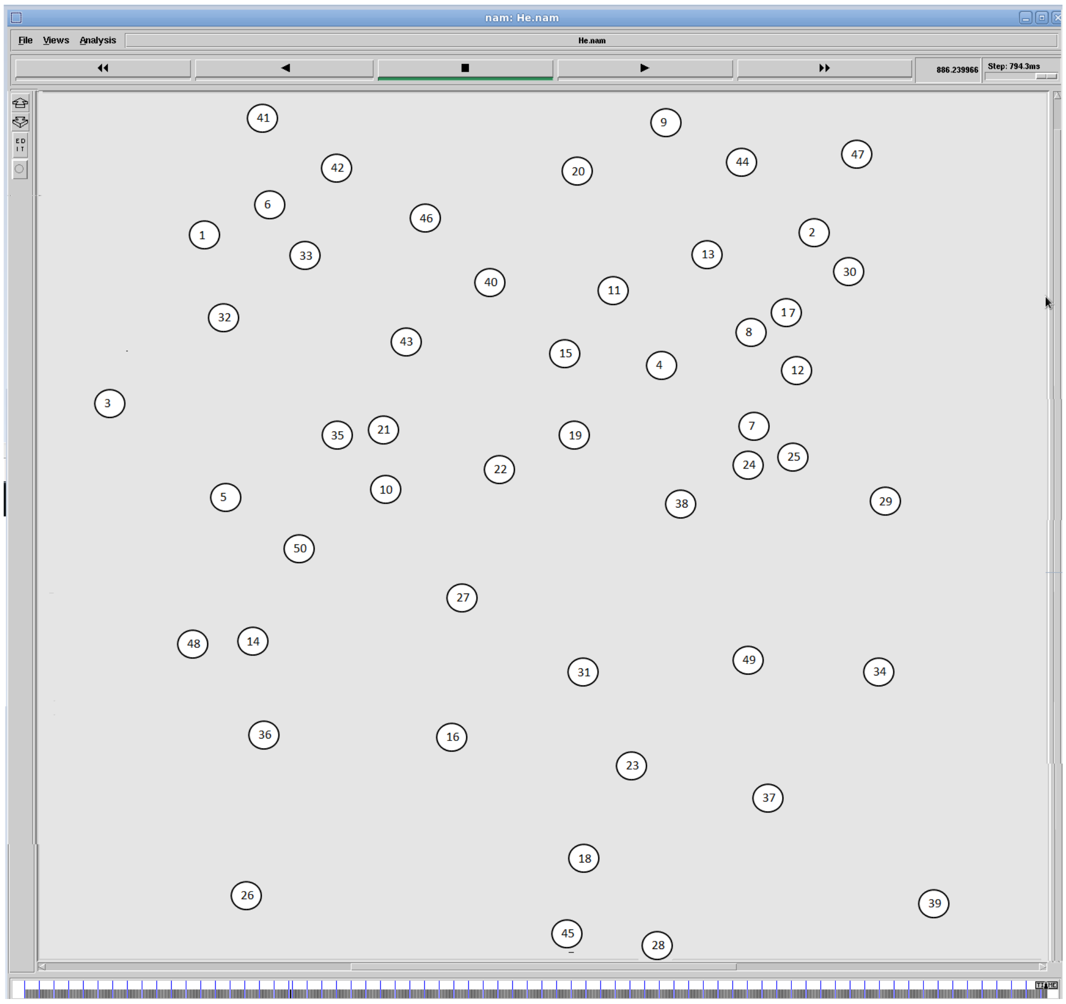


Figure 4.1: Network Topology of Scenarios 1-6

4.7.1 Scenario 1: No Intermediate Nodes

Scenario 1 is the simplest scenario, in which no intermediate node is used. In other words, the source node is directly connected to the destination node as shown in Figure 4.2. In this simulation, N_{16} is the source node, and N_{23} is the destination node.

The simulation results show that the destination node received all the data packets sent from the source node, that is the PDR value is 100%. They are delivered with 0 routing overhead, and the DLY is 0.0058 s. With regard to the 100% PDR value, this is anticipated, as the two nodes are directly connected, there is no overflow condition, and there are no packet dropping attacks. Similarly, as the two nodes directly connected neighbours, there is no need to discover paths. So, the routing overhead should be 0.

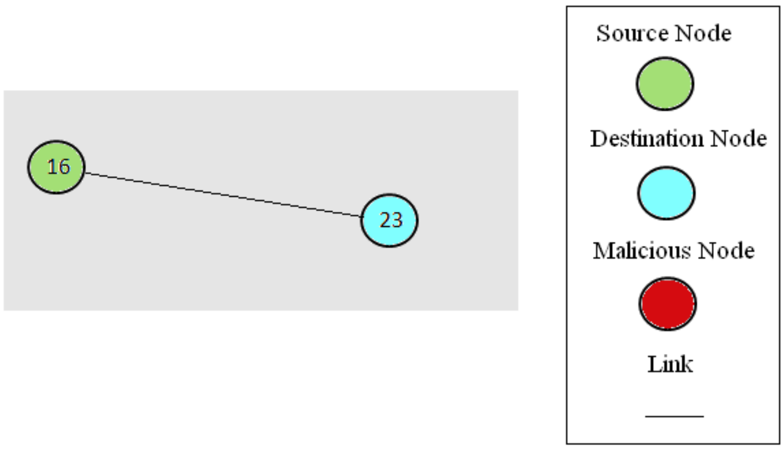


Figure 4.2: Network Topology of Scenario 1

4.7.2 Scenario 2: With Intermediate Nodes and One Path Connection the Communication Nodes

Scenario 2 has two intermediate nodes between the source and the destination nodes. This scenario requires the source node to search for a path leading to the destination node. The network topology used for this scenario is shown in Figure 4.3. In this simulation, N_{50} is the source node, and N_{49} is the destination node. There is only one path connecting the two nodes, that is $N_{50} \rightarrow N_{27} \rightarrow N_{31} \rightarrow N_{49}$.

The destination node received all the data packets sent from the source node. Data packets are delivered with 100% PDR. They are delivered with 4.65 routing overhead, and the DLY is 0.0079 s. Similar to the case in scenario 1, as there is no congestion or buffer overflow, and no malicious node, there is no packet loss. As, in this scenario, the two nodes are not connected directly, there is a need for a path discovery process, so the routing overhead has a non-zero value. Comparing the DLY in Scenario 1 and Scenario 2, it can be seen that DLY in scenario 2 is higher than the one from scenario 1, with an increase 36%. This is because, in Scenario 2, additional delays have been introduced at the two intermediate nodes when forwarding the data packets to the destination node.

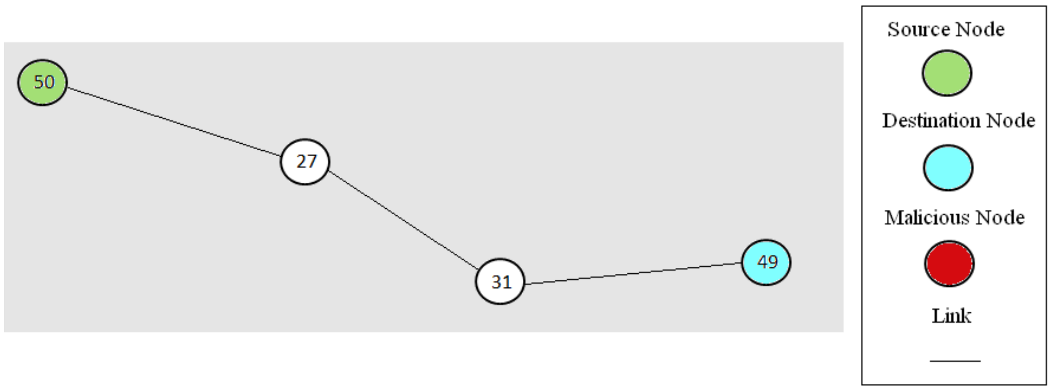


Figure 4.3: Network Topology of Scenario 2

4.7.3 Scenario 3: With Intermediate Nodes, Multi-Path Connecting the Communication Nodes, and the Required Bandwidth is 1

In scenario 3, there is more than one path connecting the source and destination nodes. The required bandwidth, BW_{req}, is 1. This scenario involves discovering one or more paths between the communication nodes and selecting one or a set of path from the discovered paths to send the data packets. The network topology used for this scenario is shown in Figure 4.4. Here, N_{40} is the source node and N_{30} is the destination node.

The simulation results show that six paths are discovered connecting the two nodes, and they are as follow:

- $N_{40} \rightarrow N_{20} \rightarrow N_9 \rightarrow N_{44} \rightarrow N_{47} \rightarrow N_{30}$, gives TRV= 6 and ABW =2
- $N_{40} \rightarrow N_{20} \rightarrow N_9 \rightarrow N_{44} \rightarrow N_2 \rightarrow N_{30}$, gives TRV= 8.2 and ABW =2

- $N_{40} \rightarrow N_{11} \rightarrow N_{13} \rightarrow N_{17} \rightarrow N_{30}$, gives TRV= 4.4 and ABW =0
- $N_{40} \rightarrow N_{11} \rightarrow N_{8} \rightarrow N_{17} \rightarrow N_{30}$, gives TRV= 4.6 and ABW =2
- $N_{40} \rightarrow N_{15} \rightarrow N_{4} \rightarrow N_{8} \rightarrow N_{12} \rightarrow N_{30}$, gives TRV=4.8 and ABW =2
- $N_{40} \rightarrow N_{15} \rightarrow N_{4} \rightarrow N_{12} \rightarrow N_{30}$, gives TRV= 8.8 and ABW =1

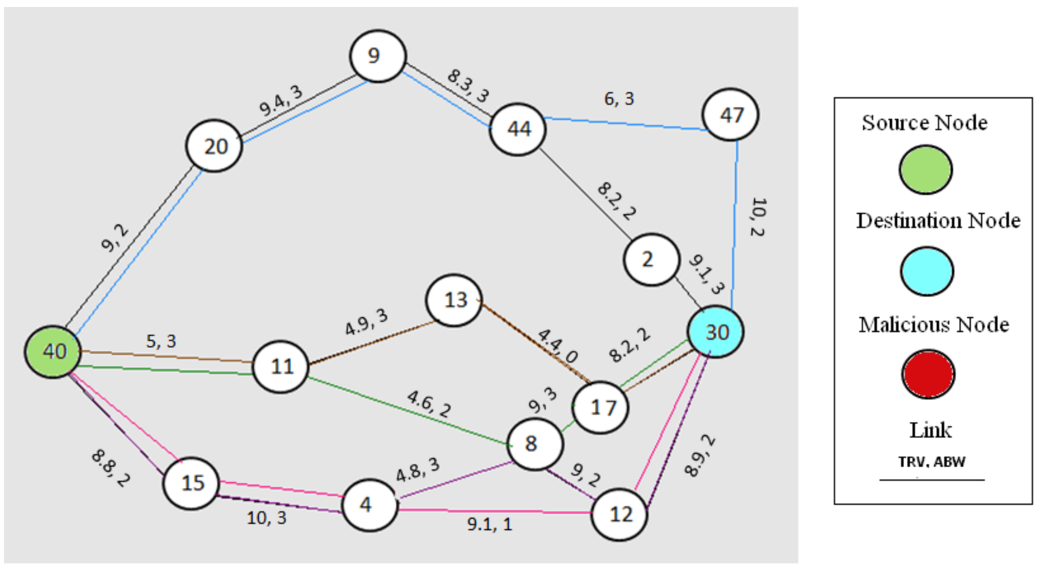


Figure 4.4: Network Topology of Scenario 3

As expected, the simulation results show that the path selected is the one with the highest trust value and sufficient available bandwidth, that is $N_{40} \rightarrow N_{15} \rightarrow N_{4} \rightarrow N_{12} \rightarrow N_{30}$. The simulation results also show that the packet delivery ratio is 100%. The destination node received all the data packets sent from the source node. They are delivered with 5.09 routing overhead, and the DLY is 0.0095 s. The PDR value of 100% is expected, as explained in scenario 2. The routing overhead is higher than the value in scenario 2, as, in this scenario, multiple path are discovered versus scenario 2 in which only one path is discovered. Discovering more paths requires more control packets being poured in the network to satisfy the number of paths required, resulting in a higher level of routing overheads. With regard to the average end-to-end packet delivery delay, the value here is 20% higher than the value in scenario 2. This increase could be explained by the fact that the nodes in the network are busier than in the case in scenario 2, as they need to handle more control packets.

4.7.4 Scenario 4: With Intermediate Nodes, Multi-Path Connecting the Communication Nodes, and the Required Bandwidth is 4

In scenario 4, multiple path are required between a pair of source and destination nodes. This is because, in *ATL-QoS*, more than one path is required if there is no single path that can satisfy the required bandwidth. This scenario involves discovering multiple path between the communicating nodes and then selecting one path or a set of paths from the discovered paths. The selected paths should satisfy the required bandwidth to send the data packets. The network topology used for this scenario is the same as in scenario 3 (shown in Figure 4.4). Here, N_{40} is the source node and N_{30} is the destination node, and $BW_{req} = 4$.

The simulation results show that the same six paths, from scenario 3, connecting the two nodes are discovered. However, here, the simulation results show that no single path can satisfy BW_{req}. In this case, the source node will select all the possible set of node-disjoint paths that collectively satisfy BW_{req}. Four such sets of paths are found, and they are listed below:

- The first set of paths consist of two paths:
  - $N_{40} \rightarrow N_{20} \rightarrow N_9 \rightarrow N_{44} \rightarrow N_{47} \rightarrow N_{30}$, gives TRV= 6 and ABW =2
  - $N_{40} \rightarrow N_{11} \rightarrow N_8 \rightarrow N_{17} \rightarrow N_{30}$, gives TRV= 4.6 and ABW =2

  For this set, the TRV_{s1} is 4.6, and the ABW_{s1} is 4.

- The second set of paths consist of two paths:
  - $N_{40} \rightarrow N_{20} \rightarrow N_9 \rightarrow N_{44} \rightarrow N_2 \rightarrow N_{30}$, gives TRV= 8.2, ABW =2
  - $N_{40} \rightarrow N_{11} \rightarrow N_8 \rightarrow N_{17} \rightarrow N_{30}$, gives TRV= 4.6 and ABW =2

  For this set, the TRV_{s2} is 4.6, and the ABW_{s2} is 4.

- The third set of paths consist of two paths:
  - $N_{40} \rightarrow N_{20} \rightarrow N_9 \rightarrow N_{44} \rightarrow N_{47} \rightarrow N_{30}$, gives TRV= 6 and ABW =2
  - $N_{40} \rightarrow N_{15} \rightarrow N_4 \rightarrow N_8 \rightarrow N_{12} \rightarrow N_{30}$, gives TRV=4.8 and ABW =2

  For this set, the TRV_{s3} is 4.8, and the ABW_{s3} is 4.

- The fourth set of paths consist of two paths:
  - $N_{40} \rightarrow N_{20} \rightarrow N_9 \rightarrow N_{44} \rightarrow N_2 \rightarrow N_{30}$, gives TRV= 8.2, ABW =2
  - $N_{40} \rightarrow N_{15} \rightarrow N_4 \rightarrow N_8 \rightarrow N_{12} \rightarrow N_{30}$, gives TRV=4.8 and ABW =2

  For this set, the TRV_{s4} is 4.8, and the ABW_{s4} is 4.

As expected, the simulation results show that the third set of paths is used as it has the highest trust value (the third and fourth sets of paths have the same trust values). The simulation results also show that the PDR is 100%. The routing overhead is 5.9, and the DLY is 0.01 s. Comparing with the results from the previous scenarios, both routing overhead and average end-to-end packet delivery delay are higher. This is as we anticipated, as multiple paths are discovered in this scenario. So, discovering more paths means the source node needs to send more control packets over the network, which means a higher level of routing overhead. Similarly, more packets injected into the network means the packets will experience longer delays.

The simulation model verification carried out so far assumes that there is no congestion or malicious nodes in the network. In the following two scenarios, we assume that the network contains malicious nodes. We compare the simulation results obtained from our *ATL-QoS* framework with the results from the *LAR* protocol [52]. The use of the *LAR* protocol in comparison has been explained in Chapter 5.

4.7.5 Scenario 5: With Intermediate Nodes, Multi-Path Connecting the Communication Nodes, the Required Bandwidth is 1, and a Malicious Node

Scenario 5 is the same as scenario 3 with a packet dropping attacker added in the best path between the communicated nodes. The network topology used for this scenario is shown in Figure 4.5. As discussed in scenario 3, multiple path are discovered between the communication nodes. In scenario 5, N_4, which is one of the intermediate node forming the path that is selected and used in scenario 3, is assigned as an attacker node. N_4 is assigned as an attacker node to further validate the simulation model of the framework. We let N_4 plays the role of a packet dropping attacker and selfish node, respectively, and collect results from the simulation runs.

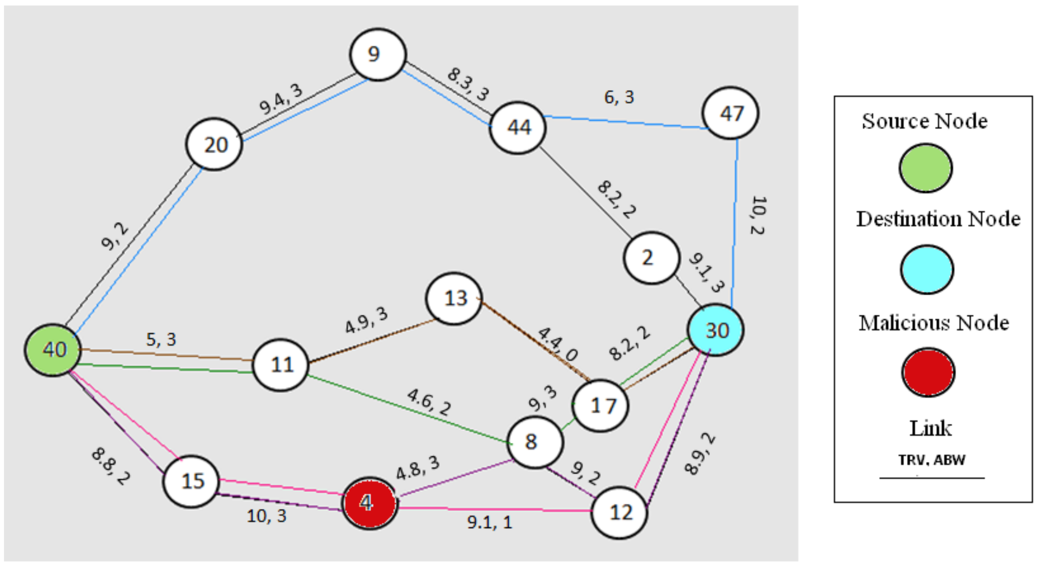


Figure 4.5: Network Topology of Scenario 5

When N_4 plays the role of a packet dropping attacker, the simulation results of the *ATL-QoS* framework show that the *PDR* is 55.46%, the routing overhead is 6.49, and the *DLY* is 0.012 s. Under the same parameter setting, the simulation results of the *LAR* protocol show that the *PDR* is 41%, the routing overhead is 0.175, and the *DLY* is 0.037 s.

From the results, it can be seen that *ATL-QoS* outperforms *LAR* in terms of *PDRs*. This is because the attacker in both situations will act normally during the path discovery phase, leading the selection of the best path. This path contains the attacker. Then, during the data transmission phase, the feedback mechanism used in *ATL-QoS* allows the destination node provide feedback and the source node to reselect a path based on the feedback. Once the new path is selected, the data will be transmitted along the new path. In the case of *LAR*, however, there is no change during the data transmission phase, i.e. data will use the same path containing the attacker in the entire phase. As *ATL-QoS* adjusts its path selection, i.e. selects more reliable paths, during the data transmission phase in respond to the feedback provided by the destination node, more data packets can be delivered to the destination node successfully, thus achieving a higher *PDR* value. Also, as a part of the path selection adjustments, the source node initiates more path discovery, pouring more control packets into the network, leading to a higher level of routing overhead than *LAR*. Using a less reliable path, in the case of LAR, leads to a longer DLY value (as observed from the simulation results).

When we use N_4 as a selfish node, the simulation results of the $ATL\text{-}QoS$ framework show that the PDR is 100%, the routing overhead is 4.13, and the DLY is 0.012 s. In the case of LAR, the simulation results show that the PDR is 41%, the routing overhead is 0.175, and the DLY is 0.037 s. From the result, it can be shown that the PDR of the $ATL\text{-}QoS$ is higher than the LAR. This is because, in this scenario, the selfish node will drop both the data and control packets. In $ATL\text{-}QoS$, this will decrease the link trust value. So, it will not be selected as the best path during the path discovery phase. In the case of LAR, data will use the same path containing the attacker in the entire phase. In $ATL\text{-}QoS$, the source node discovers multiple path, pouring more control packets into the network, leading to a higher level of routing overhead than LAR. From the simulation results, it can be observed that using less reliable path leads to a longer DLY value.

4.7.6 Scenario 6: With Intermediate Nodes, Multi-Path Connecting the Communication Nodes, the Required Bandwidth is 4, and a Malicious Node

Scenario 6 is the same as scenario 5 except that here in scenario 6 the required bandwidth, BW_{req}, is 4 (as in scenario 4). The network topology used for this scenario is shown in Figure 4.5. However, the simulation results show that there is no single path can satisfy BW_{req}. In this case, the source node will select all the possible set of node-disjoint paths that collectively satisfy BW_{req}. Four such sets of paths are found as described in scenario 4.

The simulation results of the $ATL\text{-}QoS$ framework show that the PDR is 73.4%, the routing overhead is 6.13, and the DLY is 0.01 s, whereas, the results of the LAR protocol shows that the PDR is 48%, the routing overhead is 0.024, and the DLY is 0.084 s.

In comparison with the LAR protocol, $ATL\text{-}QoS$ produces a higher PDR. This is because $ATL\text{-}QoS$ supports the use of the feedback from the destination node that allows the source node to reselect the paths in response to the feedback during the data transmission phase. In the LAR, on the other hand, the data will use the same path containing the attacker throughout the data transmission phase. As $ATL\text{-}QoS$ reselects paths in the middle of the data transmission phase, and if there are no sufficient paths in the routing table, the source node will initiate a new path discovery process. Therefore, the routing overhead introduced by *ATL-QoS* is higher than the *LAR* protocol. In terms of DLY, as *ATL-QoS* adjusts path selection dynamically ensuring more reliable paths are used throughout a data transmission phase, so the resulting DLY value is lower than the value in *LAR*.

The main difference between scenario 6 and 5 is that the former supports the use of multiple path in delivering traffic as against the single path delivery used in scenario 5. So, scenario 6 produces a higher value of PDR. In addition, comparing scenario 6 and 5, the routing overhead and DLY are marginally lower. The 16% decrease in the DLY value may be due to the use of multiple path, so the traffic load can be distributed across the multiple path, reducing overloading on the intermediate nodes and this can reduce the delay introduced by queuing.

4.8 Generating Statistically Significant Simulation Results

To produce statistically significant results, we should determine the number of simulation runs such that the simulation results are collected when the model is in its stable state. This section discusses how the number of simulation runs is determined. In addition, a number of weighted values are used in our *ATL-QoS* framework trust value calculation and the threshold value THR_{obs} that is used in the TVE method. This section also reports the simulation results on which the weight values are determined.

4.8.1 Determining the Number of Simulation Runs

A simulation result is produced by averaging the data collected from n independent simulation runs. The value for n should be sufficiently large to ensure that the result from the simulation is reliable. We use simulations to determine the number of simulation runs as shown in Figure 4.6. The figure displays the packet delivery ratios, ($PDRs$), for 40% traffic sources averaged from 1 to 100 simulation runs. The pause time used is 300 s. The average PDR results are shown with the red curve. The PDR values from each set of simulation runs are shown with the blue curve. The n values used in each set are shown in the X-axis.

It can be seen from the figure that the PDR for 5 runs is 85.19%, for 15 runs is 82.6%, for 30 runs is 82.75%, for 50 runs is 83.03%, and for 80 runs is 83.08%. In other words, the PDR value averaged over the results collected from 30 simulation runs has a confidence interval of 2.86 and a confidence level of 94%. Based on these results, a decision is made to collect simulation data over 30 simulation runs.

The confidence level is measured to ensure that results are collected after the simulation is in a stable state. It is measured based on the standard error of the mean (SEM) [96] that represents the error in predicting the mean of a distribution. SEM = Standard Deviation / sqrt (Total Simulation Runs), where the Standard Deviation is a measure of how spread out results are.

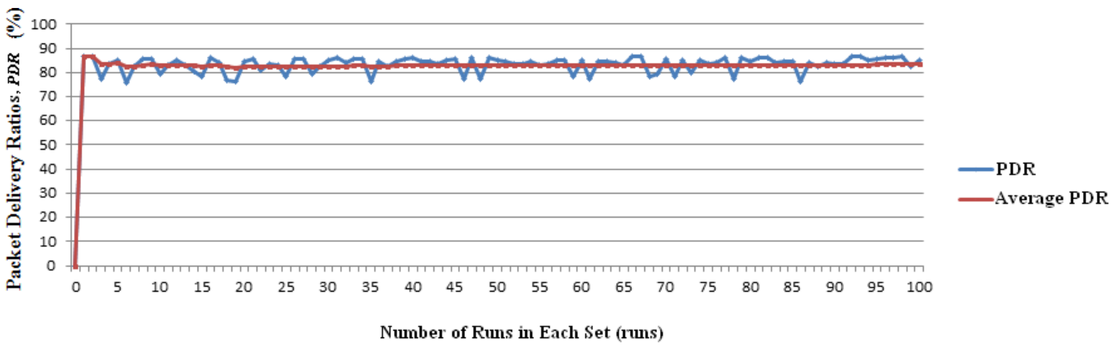


Figure 4.6: Determining the Number of Simulation Runs

4.8.2 Determining THR_{obs} Values

The simulation results, shown in Figure 4.7, are to determine an appropriate value for THR_{obs}. The THR_{obs} parameter is a threshold value used in the TVE method, (discussed in Section 4.3), to determine when to decrease or to increase the value of $TRV_{L_{ji}}^{Tcur}$.

In this simulation study, four values of the THR_{obs} parameter are investigated. They are -1, -0.5, 0 and 0.5. The investigation study is carried out with varying attacker ratios in the network. The attacker ratio of 0%, 5%, 10%, 15% and 30% are used. The simulations are conducted in a network where 40% of the nodes are source nodes, i.e. they transmit traffic loads. The pause time used is 600 s. All other simulation parameter values that are used in these three simulation studies are as specified in Section 4.6.

As shown from both figures, $THR_{obs}0$ offers the lower routing overhead in the network. Therefore, $THR_{DFB} = 0$ is chosen.

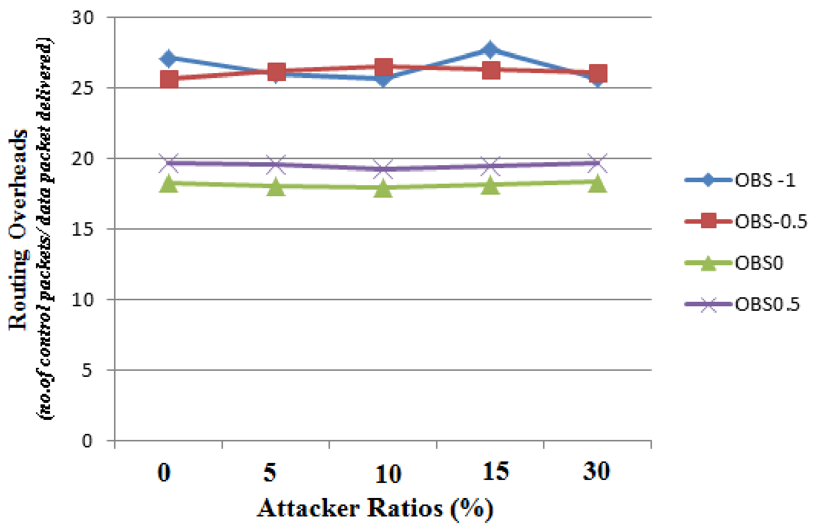


Figure 4.7: Routing Overheads vs. Network Nodes' Mobility Levels with Different THR_{obs} Values

4.8.3 Determining Weight Values

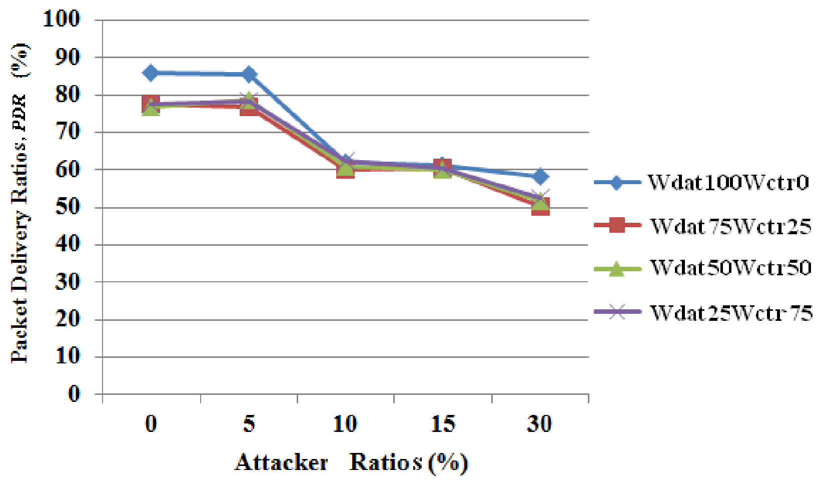
This section determines the weight values used in trust value calculation discussed in Section 4.3. According to Equation 4.2, a node's trust value is derived based on observation values, and the observation values are estimated for control packets and data packets, respectively, and different weighting can be assigned to the control and data packet related observation values respectively. This is because we have observed from literature [85, 88] that attackers may exhibit different behaviours with control and data packets. For example, in packet dropping attacks, attackers forward control packets but drop data packets. In this case, data packets are considered more important than control packets as far as trust values are concerned. So, the weights assigned to the data packets, W_{dat}, should have a higher value than the weight assigned to the control packets, W_{ctr}. In the case of selfish or overloaded nodes, both data and control packets are dropped, so data and control packets are considered equally important. Therefore, W_{dat} should have the same value as W_{ctr}. In the following, we present our simulation results to determine the validity of the weighting values assigned to data and control packets, respectively, under three cases: (a) the network only has packet dropping attackers, (b) the network only has selfish nodes, and (c) the network has both types of misbehaving nodes.

Figure 4.8 shows the PDR values versus various attacker ratios when different weighting values are used fof data and control packets. In these simulation runs, the traffic load is generated by 40% of the nodes. Attacks (packet dropping and selfish) are carried out by 0% - 30% of the nodes in the network in 5% increments.

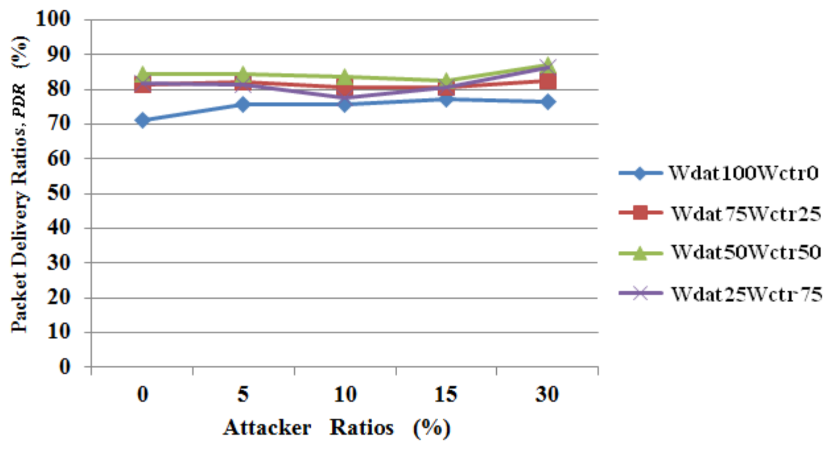
Figure 4.8(a) presents the simulation results when the network is only under packet dropping attacks. The pause time used is 600 s meaning that a low network nodes' mobility level is assumed. The reason for using a low network nodes' mobility level is to ensure that the main cause of packet loss is packet dropping attacks. That is, we want to rule out packet loss as caused by link breakage due to node mobility. The simulation results shown in the figure show that the PDR decreases as the attacker ratio increases. This observation is in line with our expectation, i.e. a higher attacker ratio means that more data packets are dropped due to attacks. Also, from the figure, we can see that the network performs best in terms of PDR when weights value assignment is $Wdat100Wctr0$ meaning that 100% weight is assigned to data packets when estimating trust values. As in packet dropping attacks, the attackers only drop data packets, so data packets should be considered more important than control packets. In other words, the data packets should be observed in this case, therefore, W_{dat} is assigned a weight of 100%.

Figure 4.8(b) shows the results when only selfish nodes are assumed. The pause time used here is 300 s meaning a high network nodes' mobility level is used. The reason for using high network nodes' mobility level is that under a high network nodes' mobility level, the network would be more likely to be congested and nodes are more likely to be selfish. When nodes are selfish, they drop both data and control packets. From the result shown in the figure, we can see that the network achieves the best PDR when $Wdat50Wctr50$ is used. In other words, under such case observing data and control packets are equally important, so W_{dat} and W_{ctr} should have equal weighting, i.e. equal importance.

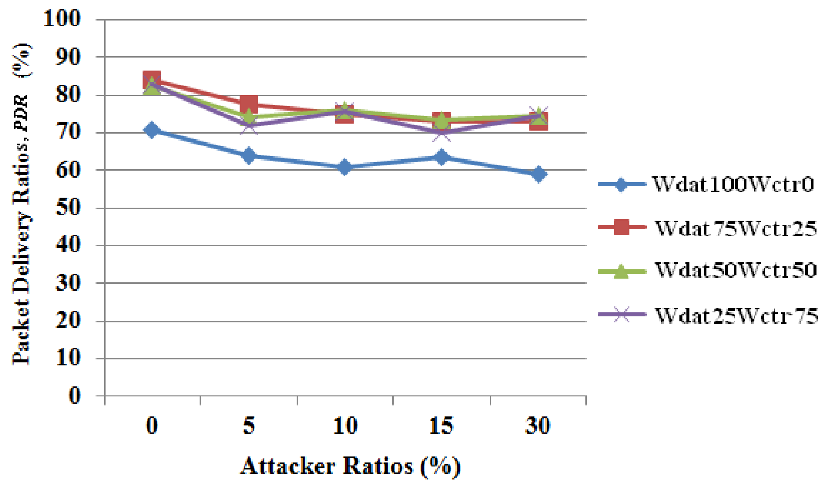
Figure 4.8(c) shows the results when both misbehaving types are assumed, i.e. 50% of the attackers are packet dropping attackers, and 50% are selfish nodes. The pause time used is 300 s meaning that a high network nodes' mobility level is used with the intention of creating overloaded nodes. The simulation results plotted in the figure show that the PDR decreases as the attacker ratio increases. A high attacker ratio means more data packets are dropped due to attacks. From the figure, we can also see that the best PDR occurs when the weights assignment is $Wdat75Wctr25$. This is because, in this setting, the packet dropping attackers only drop data packets, and the selfish and overloaded nodes drop both data



(a) Data Packets Dropping Attacks



(b) Selfish Nodes



(c) Data packets Dropping Attacks and Selfish Nodes

Figure 4.8: Determining Weight Values

and control packets. Data packets are considered more important than control packets, and W_{dat} should have a higher value than W_{ctr}. The figures also show that using the weight assignment of $Wdat100Wctr0$ produces the worst PDR. This indicates that when selfish and overloaded nodes are active, the control packets should also be observed when estimating the node trust value, and this means that a non-zero weight should be assigned to the control packet.

4.9 Chapter Summary

This chapter has presented the building blocks of *ATL-QoS* and has discussed how the framework is evaluated. These building blocks are the Trust Value Estimation (TVE) method and Available Bandwidth Estimation (ABE) method. The simulation was selected as the investigation methodology, and the Network Simulator *NS-2* was used as the simulator. The configuration of *NS-2* was described in detail. The metrics to be used for performance evaluation in the following chapters were also discussed. Simulation model validation was performed, and the number of simulation runs and weight value assignments were determined experimentally.

Chapter 5

TADL: A Trust-Aware Dynamic Location-Based Multiple Path Discovery Protocol for MANETs

5.1 Chapter Introduction

This chapter presents the design and evaluation of the Trust-Aware Dynamic Location-Based ($TADL$) multiple path discovery protocol. $TADL$ is a novel path discovery protocol for discovering multiple path with minimal bandwidth overheads in the presence of node mobility and packet dropping attacks. To discover a sufficient number of paths that are as reliable as possible, while, at the same time minimising the bandwidth cost incurred in discovering the paths, we have taken the following feature measures in the design of $TADL$. Firstly, it uses a destination location-based path discovery approach to discover paths. With this approach, route request packets are only broadcast within a defined search area in the network reducing the volume of traffic the network has to handle. Secondly, it dynamically adjusts the size of the starting search area in adaptation to the underlying network conditions. Thirdly, the protocol assesses and uses nodes' direct trust values to govern the selection of nodes in the path constructions.

The chapter is organised as follows. Section 5.2 describes the ideas used in the $TADL$ design. Section 5.3 describes the design of $TADL$. Section 5.4 describes the operations of $TADL$ protocol in source nodes, intermediate nodes, and destination nodes, respectively. Section 5.5 reports the simulation study of the protocol. Finally, section 5.6 concludes the chapter.

5.2 TADL: Ideas

This section gives the ideas used in the $TADL$ design. As mentioned in Section 3.2, the $TADL$ protocol is designed with three criteria in mind: (a) it should be able to discover multiple path between a pair of nodes (source and destination) at the lowest bandwidth cost possible; (b) the source node should be able to respond to the underlying dynamic network conditions as caused by node mobility and misbehaviour by intermediate nodes, etc; (c) while discovering paths, the protocol should be able to select the most reliable nodes to form the paths to optimise successful packet delivery ratios during data transmission phase, and do so with as low communication and computational costs (imposed on the intermediate nodes) as possible.

To meet these design criteria, four ideas are used in the $TADL$ design. The first is to use a directional location-based approach to multiple path discovery. The path discovery packets are forwarded within a search area bounded by the source and destination nodes, rather than flooding them into the whole network regardless of the locations of the two nodes. In this way, we can reduce the level of broadcast traffic injected into the network, thus reducing bandwidth overheads. There are two challenges with a traditional location-based approach. One is how to determine the search area that covers the physical location of the source and destination nodes. The other is how to determine the size of the search area and what other features should be taken into consideration when determining the size of the search area. With regard to the first challenging issue, each node stores the location information of other network nodes, collected from the received control and data packets, in its routing table. However, as nodes may be mobile the location information of the destination node, N_D, maintained by the source node, N_S, may not be correct. In this case, the source node, N_S, will predict N_D's current location using the location information stored in its routing table. This predicted location will be used to estimate the search area for the destination node. How the location of a destination node is predicted is described in the Destination Node Predicted Location (DPL) method, to be described in Section 5.3.2.1.

With regard to the second challenging issue, the size of the search area should be governed by two considerations: (a) a sufficient number of paths should be discoverable, and this number is determined by the traffic requirement; (b) upon satisfying (a), the control packets used in the path discovery should be as few as possible. Based on these considerations we have defined three distinctive sizes of the search area (small (Area-S), medium (Area-M) and large (Area-L)), and the selection of an area size is made in response to the underlying network conditions. In other words, we adjust the size of the starting search area dynamically in response to the underlying network conditions to optimise the trade-off between the number of paths that can be discovered and the level of bandwidth overhead required to discover the paths. This is our second idea used in the design.

From Area-S to Area-M and then to Area-L, the size of the search area progressively increases, the number of $RREQ$ packets poured into the network also progressively increases, and more paths may be discovered. Of course, this also implies that more bandwidth overheads will be generated. As mentioned above, we need to optimise the trade-off between the number of paths that may be discovered and the level of traffic poured into the network to discover the paths. Different network conditions may impact on the number of paths that can be discovered or used. For example, the network attacker ratio may influence the number of paths that should be discovered. Intuitively, to increase or maintain delivery reliability, the higher the attacker ratio, the more paths we should discover. To optimise the trade-off between the number of paths that can be discovered and the bandwidth overheads consumed by path discovery packets, we have designed a novel Adjusting Starting Searching Area $A2SA$ algorithm. $A2SA$ governs the selection of one of the three search area sizes (Area-S, Area-M, or Area-L) to be the starting search area size. $A2SA$ is explained in Section 5.3.2.2.

The third idea is to select the most trustworthy set of one-hop neighbouring nodes [65] and only multi-cast route request packets to this set of nodes, rather than broadcast the request packets to all the one-hop neighbouring nodes that are inside the search area. In this way, we may be able to discover more reliable (i.e. more trustworthy) paths, while reducing routing overhead. This idea is explained in the Trust-based Neighbouring Selection (TNS) algorithm presented in Section 5.3.2.3.

To use trust values to govern the selection of neighbours in path formation, the overhead incurred in trust estimation should also be kept as low as possible. For this, we have used a direct trust model [88, 66]. The direct trust model does not require trust value exchanges among nodes on the whole network scale and prevents the need for protecting and verifying any trust values received, thus costing less in terms of communication overhead and computational complexity on the intermediate nodes than the global trust model. This cost efficient measure ensures that better quality paths are discovered with minimum overhead costs. The measure is encapsulated in the Trust-based Neighbouring Selection (TNS) algorithm described in Section 5.3.2.3.

5.3 TADL: Detailed Design

This section describes the design of $TADL$ in detail. It first gives the notations used in the description of the design and then describes the design.

5.3.1 Notations

The notations used in the $TADL$ design description are summarised in Table 5.1. These notations are then used throughout the thesis.

Table 5.1: Notations Used in the $TADL$ Protocol Description

Notation	Definitions
R	Radius of a node predicted location
exs	Existing path search
prv	Previous path search
T_{exs}	The time when the existing search is carried out
T_{prv}	The time when the previous search is carried out
T_{lst}^{SD}	The last time N_S heard from N_D
SAS_{prv}	Previous search area size
$V_{T_{lst}}^{D}$	N_D mobility speed at time T_{lst}
$AvrMob_{ngb}$	The neighbouring node's mobility level
THR	THReshold value
THR_{att}	Attacker ratio threshold
THR_{mob}	Mobility level threshold
THR_t	Searching time threshold
NPS	Number of Paths
NPS_{req}	The required number of paths
NPS_{prv}	The previous search discovered number of paths
MTN	Number of Most Trusted Neighbours

5.3.2 TADL Components

This section describes the $TADL$ components and their design. Figure 5.1 shows the $TADL$ components and three of the components are $TADL$ specific. These three $TADL$ components along with the rest shown in the figure collectively perform the functions defined for $TADL$. The $TADL$ components are the Destination Node Predicted Location (DPL) method, the Adjusting Starting Searching Area ($A2SA$) algorithm, and the Trust-based Neighbouring Selection (TNS)

algorithm. The other components are the four tables, the Local Information Management component and the Best Path Selection (BPS) algorithm. The tables and the Local Information Management component have been described in detail in Chapter 3, so we have focused on describing the three $TADL$ components and the BPS algorithm.

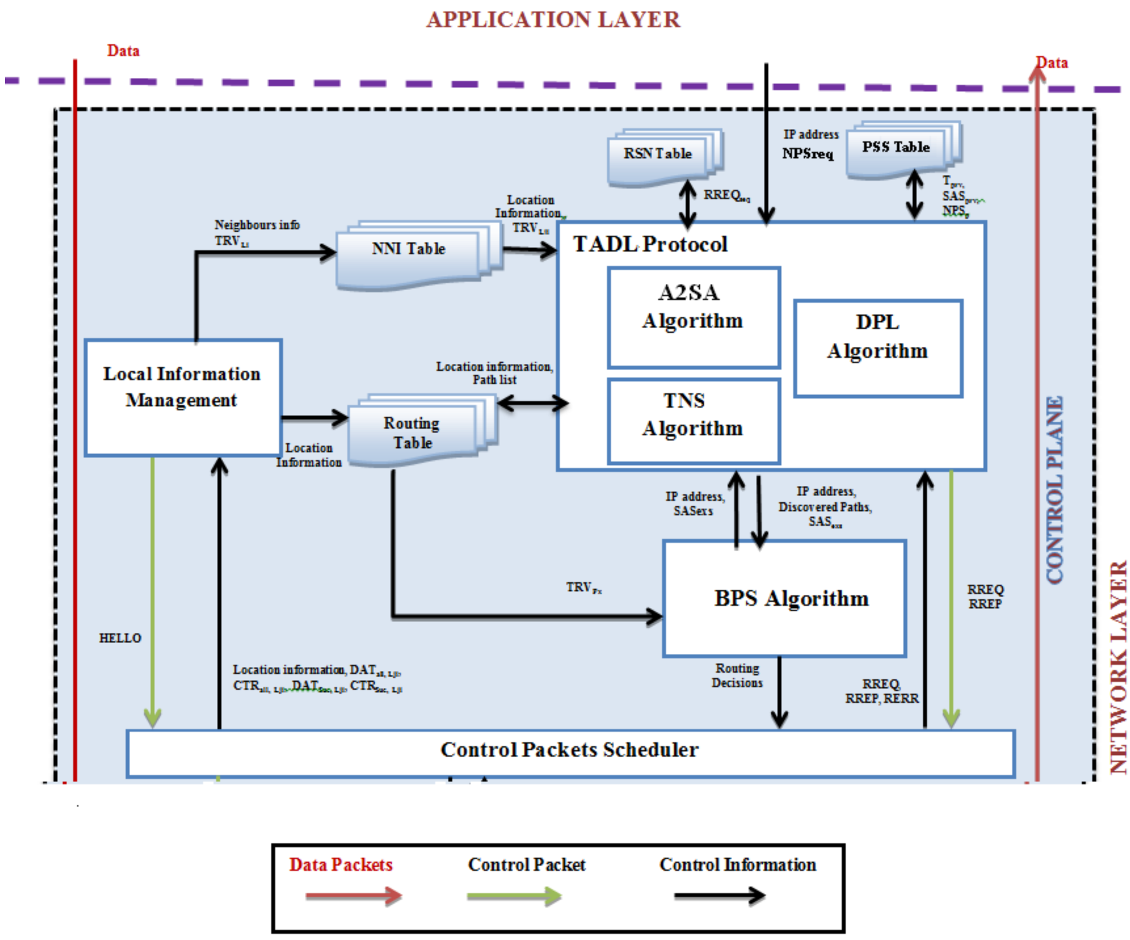


Figure 5.1: $TADL$ Components

5.3.2.1 The Destination Node Predicted Location (DPL) Algorithm

The DPL algorithm, shown in Algorithm 5.1, is used to predict the location of a destination node. The predicted location of a destination node, N_D, as mentioned in Section 3.3.2, is the region where the source node, N_S, expects N_D to be, at time T_{cur}, based on the known physical location and moving speed of N_D at time T_{lst}^{SD}, where T_{lst}^{SD} is the time when N_S received the latest location information update about N_D. The predicted location of N_D is a circular region around N_D's known physical location, $(X_D; Y_D)$, at time T_{lst}^{SD}. The radius (R) of this circle is calculated as follows:

$$R = V_{T_{lst}}^{D} \times (T_{cur} - T_{lst}^{SD}) \times SAS_{exs} \tag{5.1}$$

where $V_{T_{lst}}^{D}$ is N_D's movement speed at time T_{lst}^{SD}. SAS_{exs} is a value indicating the starting search area size (explained in Section 5.3.2.2).

Algorithm 5.1 The *DPL* Algorithm

```
function DPL($IPaddress, SAS_{exs}$)
    Read $V_{T_{lst}}^{D}$ and $T_{lst}^{SD}$ for $IPaddress$ from the routing table
    $R = V_{T_{lst}}^{D} \times (T_{cur} - T_{lst}^{SD}) \times SAS_{exs}$
    Determine $N_D$'s $predictedlocation$
     return $N_D$'s $predictedlocation$
end function
```

5.3.2.2 Adjusting Starting Searching Area (A2SA) Algorithm

A2SA implements a set of rules to govern the selection of a starting search area size. The search area, as mentioned in Section 3.3.2, is used to reduce the number of control packets injected into the network by only forwarding the control packets in this area, rather than flooding the packets to the whole network. The search area is a rectangular area that starts from the current location of N_S and includes the predicted location of N_D (explained in Section 5.3.2.1) at the current time T_{cur}, as shown in Figure 5.2. The edges of the search area are: $(X_S, Y_D + R)$, $(X_D + R, Y_D + R)$, $(X_D + R, Y_S)$ and (X_S, Y_S). Note that those edges don't wrap.

Three distinctive sizes of the search area have been defined as shown in Figure 5.3 (small (Area-S), medium (Area-M) and large (Area-L)). Area-S is the smallest one. For Area-S, the SAS_{exs} value, in Equation 5.1, is 1.

Area-M is the next-level size (medium). The size is such defined that the size of the predicted location around the destination node (called an expected location of the destination node) doubles the one used in Area-S. In Area-M, the SAS_{exs} value for Area-M, in Equation 5.1, is 2. Doubling the size of the expected zone increases the search area in which *RREQ* packets are broadcast, and this may lead more paths to being discovered.

Area-L is the largest search area that covers the entire network. In other words, when Area-L is used, *RREQ* packets will be broadcast to the entire network, defaulting to the flooding path discovery protocol. The selection and use

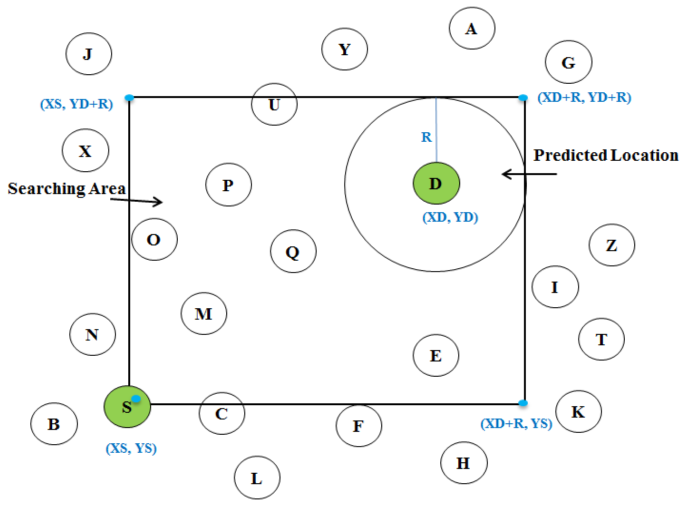
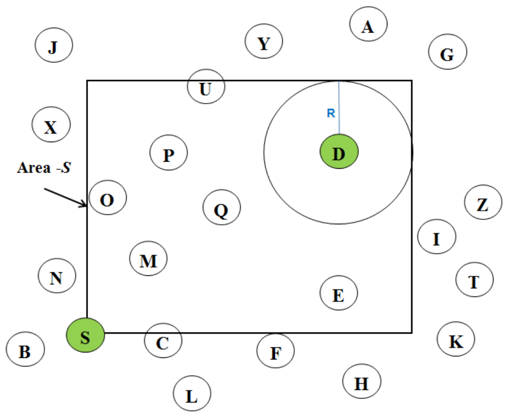


Figure 5.2: Predicted Location and Searching Area of N_D

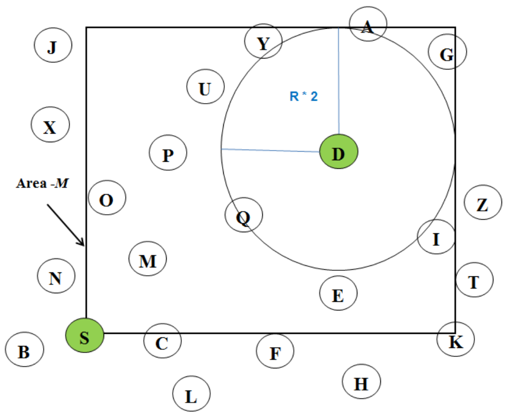
of an area size are made in response to the underlying network conditions.

The path discovery phase should start with a search area size in each path discovery attempt (one attempt is one $RREQ$ send, and hereafter referred to as one search). Then, the search area size progressively increases, if the number of discovered paths is not sufficient, i.e. the number of discovered paths should be equal or greater than the required number of paths, NPS_{req}, which we assume here is determined based on the source node application's QoS requirements. Intuitively, the selection of the search area starting size should be made based on the number of paths discovered in the previous search and NPS_{req} to be discovered by the source node in the current search. For example, if the number of discovered paths in the previous search is equal to NPS_{req}, the existing search may start from the same search area size used in the previous search. If the number of paths discovered in the previous search, SAS_{prv}, is higher than NPS_{req}, then the SAS_{exs} value used in the existing search may be the value used in the previous search, SAS_{prv}, decremented by 1. If the number of paths discovered in the previous search is less than NPS_{req}, the search area should be increased, e.g. the SAS_{exs} value used in the existing search should be incremented by 1, if the previous search area is not Area-L.

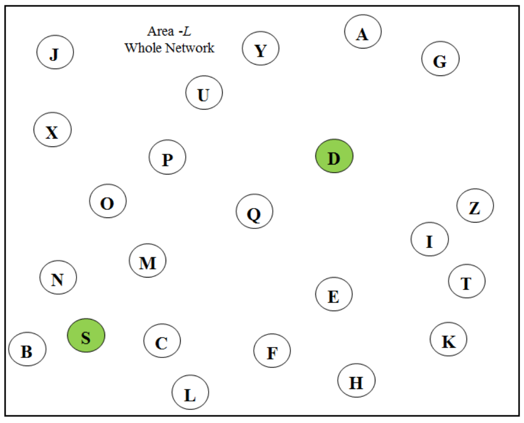
However, in MANETs, the rules that should be used to govern the selection of a search area size may be more complex than those mentioned above. The above decision rules are typically not true in a mobile MANET. There are three factors that should be considered when choosing a starting search area size to



(a) Area-*S*



(b) Area-*M*



(c) Area-*L*

Figure 5.3: Searching Area Sizes

start a search. These factors are the neighbouring nodes' attacker ratios, the neighbouring nodes' mobility level, and the time when the previous search was carried out.

Factor 1 - neighbouring nodes' attacker ratio: This is the ratio of the number of neighbouring nodes who are packet dropping attackers over the total number of neighbouring nodes. When the neighbouring node attacker ratio is high, the possibility of finding trustworthy paths in a smaller search area is lower. In this case, $A2SA$ should recommend that the source node search starts from the largest search area, Area-L, which covers the whole network. In this way, the probability of finding a sufficient number of trustworthy paths should be in this case higher.

Factor 2 - neighbouring nodes' mobility level: This is the average roaming speed of the neighbouring nodes. If the neighbouring nodes' average mobility level is high, the network topology may change more frequently and the paths discovered in the previous attempt would be more likely to become invalid. In other words, the network topology would likely be different in each path discovery attempt. In such a case, to reduce unnecessary bandwidth overheads, the source node should start with the smallest search area, Area-S.

Factor 3 - the time of the previous search attempt T_{prv}: The topology is not only affected by neighbouring nodes' mobility but also the time when the topology was last established. The longer the time elapsed since the last search operation, the more likely the topology will change. Therefore, we have specified a threshold value for the time elapsed since the last search operation. If the elapsed time is above this threshold value, the paths discovered in the previous search attempt are considered as obsolete, and even if the number of previously discovered paths is sufficient versus the required number of paths, the source node should start from the smallest search area, Area-S. The rules discussed above have been implemented into an algorithm, called $A2SA$ algorithm (shown in Algorithm 5.2).

Algorithm 5.2 The $A2SA$ Algorithm

```
function A2SA($NPS_{req}$)
    if $AttackRatio >= THR_{att}$ then
        $SAS_{exs} \leftarrow Area - L$
    else
        Read neighbouring nodes' $V_i$ from $NNI$ table
        Calculate the $AvrMob_{ngb}$
        if $AvrMob_{ngb} >= THR_{mob}$ then
            $SAS_{exs} \leftarrow Area - S$
        else
            Read $SAS_{prv}$, ($T_{prv}$), and $NPS_{prv}$ from $PSS$ table
            if $T_{cur} - T_{prv} >= THR_t$ then
                $SAS_{exs} \leftarrow Area - S$
            else
                if $NPS_{req} = NPS_{prv}$ then
                    $SAS_{exs} \leftarrow SAS_{prv}$
                else
                    if $NPS_{req} < NPS_{prv}$ then
                        if $SAS_{prv} == Area - S$ then
                            $SAS_{exs} \leftarrow Area - S$;
                        else
                            $SAS_{exs} \leftarrow SAS_{prv} - 1$;
                        end if
                    else
                        if $SAS_{prv} == Area - L$ then
                            $SAS_{exs} \leftarrow Area - L$;
                        else
                            $SAS_{exs} \leftarrow SAS_{prv} + 1$;
                        end if
                    end if
                end if
            end if
        end if
    end if
    return $SAS_{exs}$
end function
```

From the $A2SA$ algorithm (Algorithm 5.2), it can be seen that the source node first checks the neighbouring nodes' attacker ratio, if it is higher than a predefined threshold value, THR_{att}, the search area will start from Area-L. Otherwise, the source node checks the neighbouring nodes' mobility level. If the neighbouring nodes' mobility level of the neighbouring nodes is higher than a predefined threshold value, THR_{mob}, the source node will start from the smallest search area, Area-S. Otherwise, if the average mobility is low, the source node compares the time of the previous search, T_{prv} with the current time, T_{cur}. If the difference between the two times is higher than a predefined threshold, THR_t, the source node starts the search from Area-S. Otherwise, the source node checks the number of paths discovered in the previous search, NPS_{prv}. Here, there are three further possibilities:

1. If NPS_{req} is equal to NPS_{prv}, then the existing search should start from the same search area size as the size used in the previous search.

2. If NPS_{req} is less than NPS_{prv}, then the existing search area should start from the search area size that is one level lower. For example, if the previous search area is Area-M or Area-S, the next search area used should be Area-S. If the previous search area is Area-L, the next starting search area size should be Area-M.

3. If NPS_{req} is higher than NPS_{prv}, the discovery process will start from a search area size that is one level higher. For example, if the last search area is Area-S, the next starting search area size should be Area-M. If the last search area is Area-M or Area-L, the next starting search area size should be Area-L.

The THR_{att}, THR_{mob}, and THR_t values are tested experimentally in Section 5.5.1.

5.3.2.3 Trust-based Neighbouring Selection (TNS) Algorithm

In a path discovery phase, each node uses the TNS algorithm to select a set of the most trusted neighbours in the destination node search area, and then send the $RREQ$ packet to the neighbours in this set. As shown in Algorithm 5.3, N_S defines N_D's search area based on the predicted location of N_D. Then, based on the link trust values, N_S ranks its neighbouring nodes that are in the destination node search area and selects a set of the Most Trusted Neighbours (MTN) from the top of the list. The neighbouring nodes' trust values are obtained from the NNI table maintained in the node. The value of MTN is set to 3 nodes based on [65]. In [65], their experiments were carried out using different values for MTN, these values are 1, 2, 3, and 4 nodes. The simulation results in [65] show that

PDR increases when the MTN value increases and that the PDR is almost the same when the MTN value set to 3 and 4 nodes. Also, the routing overheads increase when the MTN value increases. So to avoid the unnecessary increase in routing overhead and, at the same time, to have the best MTN value, the MTN value selected is 3 nodes.

Algorithm 5.3 The TNS Algorithm

```
function TNS($PredictedLocation$)
    Determine $N_D$'s search area based on $N_D$'s $PredictedLocation$
    Read $TRV_{L_{ji}}$ for each neighbouring node from the $NNI$ table
    Rank neighbours that are inside $N_D$'s search area based on $TRV_{L_{ji}}$
    $SelectedNeighbours \leftarrow TopMTNNeighbours$
     return $SelectedNeighbours$
end function
```

5.3.2.4 Best Path Selection (BPS) Algorithm

The BPS algorithm is used to select the most trustworthy path among the discovered paths, to forward traffic to the destination node. It is executed after the path discovery phase is completed. As shown in Algorithm 5.4, given a required number of paths, NPS_{req}, which we assumed here is determined based on the source node application's QoS requirements, the source node, N_S, will check the number of discovered paths. If the number of discovered paths is above or equal to NPS_{req}, N_S will calculate a trust value, TRV_{P_x}, for each of the discovered paths. TRV_{P_x} is the minimum value of the trust values of all the links along that path. This calculation is based on the weakest link principle. The trust values of all the links along that path are stored in the path list in the routing table. Based on the trust values of the paths, N_S ranks the paths and selects the path with the highest trust value from the top of the list, and uses it to transmit the traffic. Otherwise, if the number of discovered paths is less than NPS_{req}, a new $RREQ$ will be transmitted in a larger search area than the one just used. If the previously used search area is already the largest area, Area-L, then N_D is considered to be an unreachable destination.

Algorithm 5.4 The *BPS* Algorithm

```
procedure BPS($IPaddress, DiscoveredPaths, SAS_{exs}, NPS_{req}$)
    do
        if The number of $DiscoveredPaths >= NPS_{req}$ then
            Calculate $TRV_{Px}$ for each path
            Rank the paths based on $TRV_{Px}$
            Select $TheBestTrustedPath$ that has the higher Trust value
            Transmit traffic over $TheBestTrustedPath$
            Exit
        else
            if $SAS_{exs} ==$ Area-$L$ then
                $N_D$ is an unreachable destination
                Exit
            else
                $SAS_{exs} \leftarrow SAS_{exs} + 1$
                $DiscoveredPaths \leftarrow$ TADL-S($IPaddress, SAS_{exs}$)
            end if
        end if
    while True
end procedure
```

5.4 TADL Protocol Operations

This section describes the operations of the $TADL$ protocol in discovering multiple path. The operations of the protocol differ in term of node types in which it is executed. There are three node types in $TADL$: source nodes, intermediate nodes and destination nodes. So the operations are also classified into $TADL$ source nodes operations, $TADL$ intermediate nodes operations, and $TADL$ destination nodes operations. This section describes these operations.

5.4.1 The TADL Source Node Operations

The source node component searches the network to find paths from this node, N_S, to the destination node, N_D. When N_S, has traffic to send, it first searches its routing table to see if there is already a sufficient number of paths to N_D. If yes, N_S selects and uses the most trusted path from the paths in the routing table to transmit the traffic. If not, a path discovery phase is initiated by invoking the $TADL$ protocol as follows. In this phase, N_S searches the network to find paths linking to N_D. As shown in Algorithm 5.5, N_S reads N_D location information from its routing table. If N_S does not find N_D's location information, N_S defaults to use the basic broadcast method to discover paths. In this case, N_S and all the intermediate nodes that receive the $RREQ$ packet will broadcast the $RREQ$ packet to all the neighbouring nodes until N_D is reached. However, if N_S can find N_D's location in its routing table, N_S invokes the $A2SA$ algorithm (as explained in Section 5.3.2.2) to calculate the starting search area size. Then, N_S invokes the (DPL) algorithm (described in Section 5.3.2.1) to predict the current location of a destination node. The prediction location of the destination node is used to define N_D's search area. N_S then invokes the TNS algorithm (described in Section 5.3.2.3) to select the most trusted neighbours in the destination node search area. N_S then sends the $RREQ$ packet only to these neighbours in this set.

Upon the transmission of a $RREQ$ packet, N_S initiates a route discovery timer ($RDTimer$). By the expiry of this timeout interval, if N_S does not receive any $RREP$ packets, a new $RREQ$ will be transmitted in a larger search area than the one just used (as explained in the $A2SA$ algorithm in Section 5.3.2.2). If the previously used search area is already the largest area, Area-L, then N_D is considered to be an unreachable destination. Otherwise, if, by the expiry of $RDTimer$, N_S receives $RREP$ packet(s), N_S extracts the paths from the $RREP$ packets. N_S then invokes BPS algorithm (explained in Section 5.3.2.4) to select the most trusted path from the discovered paths. N_S then sends the data traffic via the selected path to N_D.

5.4.2 The TADL Intermediate Node Operations

Intermediate nodes, upon the receipt of a $RREQ$ packet, do not generate any response back to the source node. Rather, they forward the $RREQ$ packet in the direction of the destination node until the destination node is reached. As shown in Algorithm 5.6, the operations are as follows. An intermediate node, upon receiving a $RREQ$ packet, appends the values of the $RREQ$ attributes associated to this intermediate node in the $RREQ$ header. These attributes values are the intermediate node's IP address, its location information, and the trust values of the links connecting this node to the set of selected neighbours (i.e. MTN downstream neighbours). The MTN neighbours are selected as follows. The node invokes the DPL algorithm (described in Section 5.3.2.1) to predict the current location of a destination node. It then invokes the TNS algorithm (described in Section 5.3.2.3) to select the most trusted neighbours in

Algorithm 5.5 The $TADL$ Source Node Operations

```
function TADL-S($IPaddress, SAS_{exs}, NPS_{req}$)
    Search the routing table for $N_D$'s location information)
    if D's location information is not found then
        $SAS_{exs} \leftarrow Area - L$
        $SelectedNeighbours \leftarrow Allneighbours$
    else
        if $SAS_{exs}$== Null then      ▷ First time $TADL$ is executed during this
path discovery phase
            $SAS_{exs} \leftarrow$ A2SA($NPS_{req}$)
        end if
        do
            $PredictedLocation \leftarrow$ DPL($IPaddress, SAS_{exs}$)
            $SelectedNeighbours \leftarrow$ TNS(Predicted Location)
            Send $RREQ$ to the $SelectedNeighbours$
            Initiate a $RDTimer$ timer
            Wait for a $RREP(s)$ from $N_D$
            if $RREP$(s) packets arrived then
                Read $DiscoveredPaths$ from the $RREPs$ packet
      return $DiscoveredPaths$
            else
                if $SAS_{exs}$ == Area-$L$ then
                    $N_D$ is an unreachable destination
                    $DiscoveredPaths \leftarrow Null$
      return $DiscoveredPaths$
                end if
                $SAS_{exs} \leftarrow SAS_{exs} + 1$
            end if
        while True
    end if
end function
```

the destination node search area. The intermediate node then sends the *RREQ* packet to this set of neighbours. The *TADL* intermediate node operations will be called the *SC-MNPD* method from the next chapter.

Algorithm 5.6 The *TADL* Intermediate Node Operations

```
procedure TADL-I($RREQ$)
    Add node's information to the path list in the $RREQ$ packet
    Read $IPaddress$ and $SAS_{exs}$ from the $RREQ$ packet
    $PredictedLocation \leftarrow$ DPL($IPaddress, SAS_{exs}$)
    $SelectedNeighbours \leftarrow$ TNS(Predicted Location)
    Send $RREQ$ to the $SelectedNeighbours$
end procedure
```

5.4.3 The TADL Destination Node Operations

As shown in Algorithm 5.7, when the destination node receives a *RREQ* packet, it constructs a *RREP* packet, and copies its current location information (i.e. its physical location and roaming speed values) and the entire path carried in the *RREQ* header into the *RREP* header. The destination node then sends the *RREP* to the source node via the reverse of the path.

Algorithm 5.7 The *TADL* Destination Node Operations

```
procedure TADL-D($RREQ$)
    Construct a $RREP$ packet
    Writes the node location information and the entire path into the $RREP$
    Send the $RREP$ to the source node via the reverse of the path
end procedure
```

5.5 Simulation Study

The performance of *TADL* is evaluated using a simulation study. We first determine the threshold values used in *TADL* via simulation. Then we analyse the study results of the *TADL* protocol and compare the results against those from *LAR* [52], the path discovery protocol most relevant to *TADL*.

5.5.1 Determining Thresholds

The three threshold values THR_{att}, THR_{mob}, and THR_t are used to adjust the starting search area size in the investigation of $TADL$ protocol, as described in Section 5.3.2.2. The THR_{att} , THR_{mob} and THR_t parameters are also used in $A2SA$ algorithm to determine the starting search area size. The starting search area adjustments are done to reduce the routing overhead introduced in discovering paths. Choosing the wrong threshold values can increase the routing overhead and lower the protocol performance.

The simulations are conducted in a network where 40% of the nodes are source nodes, i.e. they transmit traffic loads. The attacker ratio used is 0%. The pause time used is 600 s. These parameter values are set to reduce the effects of other dynamic factors such as traffic loads, and network nodes' mobility levels so that the effects of the threshold values on the protocol performance can better be observed and understood. All other simulation parameter values that are used in these three simulation studies are as specified in Section 4.6.

5.5.1.1 Determining THR_{att} Value

The first simulation results, shown in Figure 5.4, are to determine an appropriate value for THR_{att}. THR_{att} is a threshold value used in the $A2SA$ algorithm for *Factor 1 - neighbouring nodes' attacker ratios* (discussed in Section 5.3.2.2) to determine the level of the neighbouring nodes' attacker ratios. A source node starts a path discovery process with the largest search area size, Area-L, that covers the whole network, if the neighbouring nodes' attacker ratio level is above or equal to THR_{att}, whereas, if the neighbouring nodes' attacker ratio level is lower than THR_{att}, the starting search area size used will depend on the neighbouring nodes' mobility level.

In this simulation study, we only activate Factor 1 to monitor its effect on the routing overhead introduced in the network. Five values of the THR_{att} parameter are investigated. They are 0%, 5%, 10%, 15% and 30%, these values are the same percentage of the attacker ratios used in the simulation. The investigation study is carried out with varying attacker ratios in the network. The attacker ratio of 0%, 5%, 10%, 15% and 30% are used.

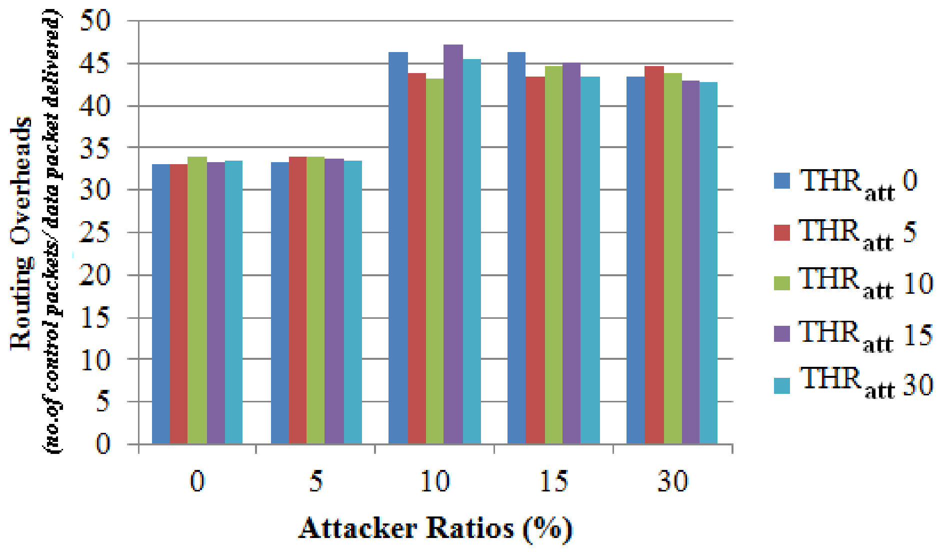


Figure 5.4: Routing Overheads vs. Network Nodes' Mobility Levels with Different THR_{att} Values

At first, the zero THR_{att} value is used to observe what is the critical attacker ratios point at which we should adjust the starting search area size. Then, we applied different THR_{att} values to examine their effects on the use of different starting search area size. From the results in Figure 5.4, we can see that the THR_{att} value doesn't appear to affect the overhead. However, as the overhead exhibits a marked increase when the attacker ratio reaches between 5% and 10%, so we believe, at this point, the search area size should be enlarged, therefore we have chosen $THR_{att} = 10\%$ as the threshold value.

5.5.1.2 Determining THR_{mob} Value

The second simulation results, shown in Figure 5.5, are to investigate an appropriate value for THR_{mob}. THR_{mob} is a threshold value that is used in the *A2SA* algorithm as *Factor 2 - neighbouring nodes' mobility level* (discussed in Section 5.3.2.2). We need this average neighbouring nodes' mobility level threshold value, THR_{mob}, if the neighbouring nodes' average mobility level is higher than THR_{mob}, the source node will start route discovery broadcasting in the smallest search area, Area-S. However, if the neighbouring nodes' average mobility level is lower or equal to THR_{mob}, the starting search area size used will depend on the time elapsed since the last search operation.

In this simulation study, we only activate Factor 2 to monitor the effect of node mobility on the routing overhead introduced into the network. Five values of THR_{mob} are investigated. They are 1m/s, 5m/s, 10m/s, 15m/s and 20m/s. The justification for using these values is that the mobility speed used in the simulation is set between 1m/s and 19m/s. The study is carried out with varying network nodes' mobility levels. The pause time values used are 900 s, 600 s, 300 s, and 0 s, respectively. To observe the THR_{mob} values that are measured in m/s, we calculate the average mobility speed from the pause time using Equation 5.2.

$$Average\_Mobility\_Speed = \frac{Max\_Mobility\_Speed + Min\_Mobility\_Speed}{2} \tag{5.2}$$

Based on the equation, when the network nodes' pause time is zero s, the average mobility speed is 10m/s. When the network nodes' pause time is 300 s pause time, which means that the nodes are mobile for two-third of the simulation time (the simulation time was set to 900 sec), the average mobility speed, in this case, is 10 × (2/3)= 6.67m/s. Similarly, when the network nodes' pause time is 600 s, the average mobility speed is 10 × (1/3) = 3.33m/s, as, in this case, the nodes are mobile for one-third of the simulation time. If the network nodes are stationary in a 900 s pause time, the average mobility speed is 0m/s.

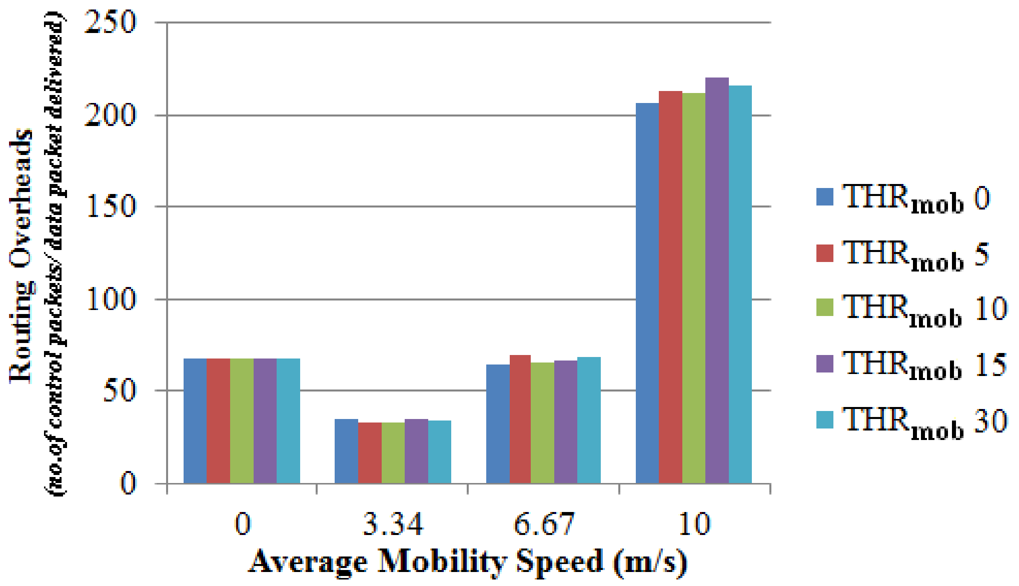


Figure 5.5: Routing Overheads vs. Network Nodes' Mobility Levels with Different THR_{mob} Values

At first, the zero THR_{mob} value is used to observe what is the critical average mobility speed point at which we should adjust the starting search area size. Then, we applied different THR_{mob} values to examine their routing overheads. As shown in Figure 5.5, we can see that the THR_{mob} value doesn't appear to affect the overhead. However, as the overhead exhibits a marked increase when the average mobility speed reaches between 6.67m/s and 10m/s, so we believe, at this point, the search area size should be enlarged, therefore we have chosen THR_{mob} = 10m/s as the threshold value.

5.5.1.3 Determining THR_t Value

The third simulation results are shown in Figure 5.6. This investigation is to determine an appropriate value for THR_t. THR_t is a threshold value used in the *A2SA* algorithm for *Factor 3 - the time of the previous search attempt* T_{prv} (discussed in Section 5.3.2.2). If the time elapsed since the last search attempt is higher than THR_t, it means that the paths sought previously are more likely to be invalid. So in this case, the source node starts from the smallest search area, Area-S, otherwise, if the time elapsed since the last search attempt is less than or equal to THR_t, the starting search area size used will depend on the number of discovered paths from the previous search.

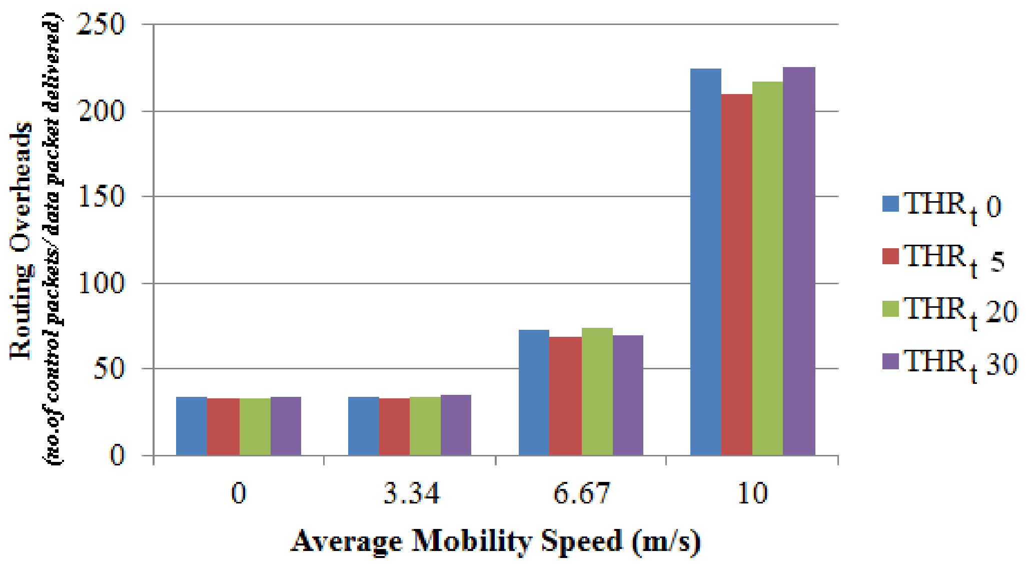


Figure 5.6: Routing Overheads vs. Network Nodes' Mobility Levels with Different THR_t Values

In this simulation study, we only activate Factor 3 to monitor its effect on the routing overhead introduced in the network. Four values of the THR_t parameter are investigated. They are 0 s, 5 s, 20 s, and 30 s. The pause time values used are of 900 s, 600 s, 300 s, and 0 s. As described in Section 5.5.1.2, these pause times are equivalent to average mobility speed of 0m/s, 3.34m/s, 6.67m/s, and 10m/s, respectively.

THR_t is dependent on mobility speed, the higher the mobility speed, the smaller the THR_t value should be. As shown in Figure 5.6, the difference caused by THR_t are not significant, so we chose a small value for THR_t to be on the safe side, i.e. a value of $THR_t = 5$ s is used.

5.5.2 Simulation Results and Discussions

This section investigates the performance of the $TADL$ protocol. The purpose is to examine the effectiveness of the ideas embedded in the design of the $TADL$ protocol in supporting traffic delivery in the presence of node mobility and packet dropping attacks. These ideas include the use of a directional location-based approach, the adjustment of the starting search area size dynamically in response to the underlying network conditions, and the use of a direct trust model to govern the selection of neighbours in path formation. The investigations are carried out by comparing the performance of the $TADL$ protocol against that of the LAR protocol (explained in Section 2.2.2) in terms of routing overheads, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$).

LAR is the most relevant protocol to $TADL$. However, as $TADL$ has novel features, there are three foundational differences between the two protocols. Firstly, $TADL$ can dynamically adjust the size of the starting search area in which path discovery packets are broadcast, whereas, LAR uses two size zones: Area-S and Area-L. LAR starts the search with Area-S and if there is no $RREP$ returned it increases to Area-L. Secondly, $TADL$ selects and uses paths that are more trustworthy. The trust value of a path is calculated based on the trust values of the links forming the path. LAR does not have this path selection functionality. Rather, LAR uses the first discovered path. Thirdly, in $TADL$, the intermediate node allows multiple copies of a $RREQ$ to pass through it with no extra checks, rather than avoids path-looping. This allows $TADL$ to discover all the possible paths during the path discovery phase. In LAR, however, an intermediate node only forwards a $RREQ$ packet once, filtering out any duplicated $RREQs$, which also avoids path-looping. In other words, after receiving a $RREQ$, the intermediate node checks the sequence number carried in the $RREQ$. If the sequence number was not seen before, the intermediate node will accept the $RREQ$. Otherwise, it will discard the packet. The implementation of LAR in *NS-2* is adopted from [55].

We study the effects of varying network node mobility levels, traffic loads and attacker ratio levels on the routing overheads, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$). In this chapter, the $DLYs$ are represented in log scale. Three sets of simulation results are presented and analysed. In set 1, the effect of different pause times on the $TADL$ performance are investigated. The pause times are used to reflect different network nodes' mobility levels. These are 0, 300, 600, and 900 s. The 0 s pause time means network nodes move continuously. The use of 0 to 300 s pause time simulates a highly mobile network. The use of the 600 s pause time simulates a medium level mobile network. For a stationary network, the pause time of 900 s is used.

In set 2, the effects of traffic loads on the $TADL$ performance are investigated. The traffic loads are generated by 20%, 40%, 60%, 80%, and 100% of the nodes respectively. 20% of source nodes are used to simulate a lightly loaded network. 40% and 60% of source nodes are used to simulate a medium loaded network. 80% and 100% of the source nodes are used to simulate a highly loaded network.

In set 3, the effects of packet dropping attacks are investigated. The investigation is carried out by setting 0% - 30%, in 5% increments, of the nodes in the network as attackers.

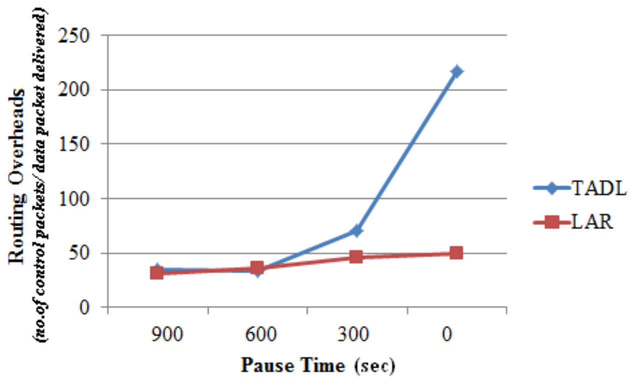
In the simulation results presented in this chapter, the number of required paths, NPS_{req}, setting should be identical to LAR, so, this parameter setting used in $TADL$ is set to one. This is because in LAR only one path is used, and to make a fair comparison between $TADL$ and LAR. All other simulation parameter values used in the three sets of simulation runs are set as specified in Section 4.6. The LAR traffic and results are marked LAR and the $TADL$ traffic and results are marked $TADL$ in the Figures 5.7, 5.8, and 5.9. Each result is obtained by averaging 30 simulation runs and the confidence level is $<= 2.86$.

5.5.2.1 TADL vs. LAR: Varying Network Node Mobility Levels

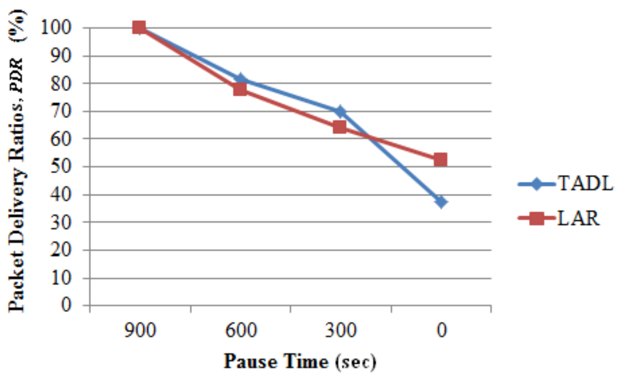
Figure 5.7 examines the effect of changing network nodes' mobility levels on the routing overheads, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$). In this set of results the traffic load used is that 40% of the nodes are source nodes and the attacker ratio is 0%.

Figure 5.7(a) shows the routing overheads versus network nodes' mobility level changes for both protocols. From the results, we can make two observations. The first observation is that the routing overheads increase as the network nodes' mobility levels increase. This is because, when the node mobility level increases, link breaks will happen more frequently. When a link breaks, packet delivery will fail. For both protocols, if a packet is not delivered due to link breaks caused by node mobility, the sending node will return a $RRER$ packet to the source node. For each $RRER$ packet received, the source node will start a new path discovery phase if the source node cannot find sufficient alternative path(s) in its routing table. This results in more control packets being generated. The second observation is that when the network nodes' mobility level is low (i.e. 600 to 900 s pause time), $TADL$ and LAR perform similarly. However, when the network nodes' mobility level is high (i.e. 0 and 300 s pause time), the routing overheads in $TADL$ increase sharply, whereas, the increase for the LAR protocol is very little, particularly, when the nodes are always mobile, i.e. when a 0 s pause time is used. $TADL$ generates excessive routing overheads. This can be explained as follows. In a highly mobile network, using small search area size, Area-S, will lead to fewer routes being discovered. If these routes become invalid quickly and frequently due to mobility, more discoveries will be initiated in a larger search area, leading to a higher level of routing overhead than LAR. In addition, in $TADL$, multiple copies of each $RREQ$ packet are allowed to pass intermediate nodes, whereas, with LAR, only one copy of each $RREQ$ is forwarded. This further amplifies the routing overhead in $TADL$. Therefore, one of the lessons learned from these investigations is that the dynamic adjustment measure used in $TADL$ is not effective or may be counterproductive in a highly mobile network.

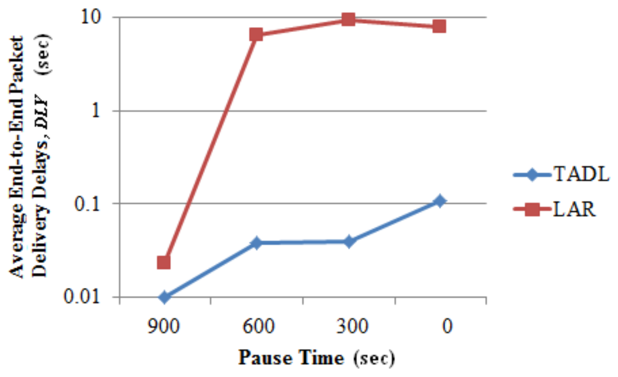
Figure 5.7(b) shows the packet delivery ratios versus network nodes' mobility level changes for both protocols. It can be seen from the figure that the $PDRs$ decrease as the network nodes' mobility level increases and the decreases are similar for both protocols except for the highest level of mobility where the PDR of $TADL$ drops more significantly than that of LAR. The decrease in the case of a



(a) Routing Overheads vs. Network Nodes' Mobility Levels



(b) Packet Delivery Ratios vs. Network Nodes' Mobility Levels



(c) Average End-to-End Packet Delivery Delays vs. Network Nodes' Mobility Levels

Figure 5.7: Routing Overheads, Packet Delivery Ratios and Average End-to-End Packet Delivery Delays vs. Network Mobility Levels for *TADL* and *LAR*

high network nodes' mobility level means more links break, and, as a result, more path discoveries are needed, which leads to more control packets being generated to discover alternative paths. This will increase the chance of the network being congested, which can lead to packet drops. From the figure, we can also see that in a low and medium network nodes' mobility level network, the PDR of $TADL$ is slightly better than that of LAR. This may be due to the fact that $TADL$ selects more reliable paths, and the network nodes' mobility level is low, it is likely that the selected path will be valid for the entire data transmission phase. However, when the network nodes' mobility level is the highest, LAR outperforms $TADL$ as, in this case, LAR produces less routing overhead as shown in Figure 5.7(a).

Figure 5.7(c) shows the average end-to-end packet delivery delays, ($DLYs$), versus network nodes' mobility level changes. From the figure, we can observe that DLY increases as the network nodes' mobility level increases. This is consistent with the results in Figure 5.7(a). A high network nodes' mobility level will lead to more links breaks, and thus more packet retransmissions. This will increase the chance of the network being congested, which in turn will increase the queuing and process time, so the data packets will take more time at the intermediate nodes to be served. Another observation from the figure is that the DLY of $TADL$ significantly outperforms that of LAR with the biggest difference in delays being 9.3ms at 300 s pause time. This can be explained as follows. Firstly, unlike LAR which only discovers a single path in each path discovery phase, $TADL$ discovers multiple path. When a path is broken, it is more likely, in the case of $TADL$, for the source node to find alternative paths in its routing table, which can be used immediately without the need for starting a new path discovery phase. This can reduce the delay in packet deliveries. Secondly, unlike LAR which involved the intermediate nodes in the path selection process, $TADL$ did not involve any intermediate node in any decision making process; intermediate nodes simply accept and pass all the $RREQ$ packets they receive. Thus, $TADL$ imposes less computational costs on intermediate nodes, and this can reduce the queuing and process time at these nodes, so the data packets take less time to be served at the intermediate nodes.

5.5.2.2 TADL vs. LAR: Varying Traffic Loads

We have also examined varying levels of traffic loads on routing overheads, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$).

In this set of figures the attacker ratio used is 0% and the pause time is 600 s, these values are used to focus on the effect of varying traffic loads in the network.

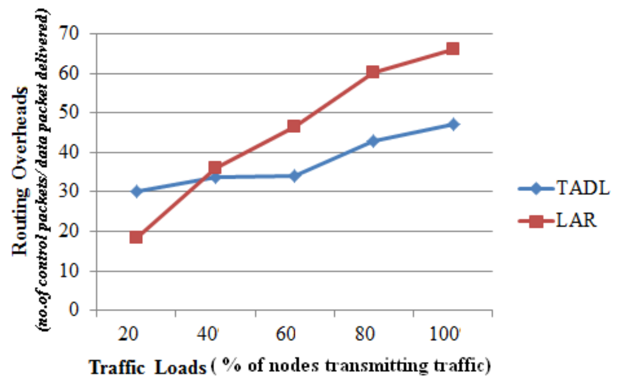
Routing overheads versus traffic loads for $TADL$ and LAR protocols are shown in Figure 5.8(a), it can be seen that, for both protocols, when the network load increases, routing overheads increase. However, the increase in $TADL$ is a lot milder than that of LAR. For example, when the traffic load is 100% (i.e. under the heavy traffic load condition), the routing overhead generated by $TADL$ is 34.8% lower than LAR. Obviously, more traffic means more packet transmissions, which requires more control packets to discover more paths to deliver them. Thus, more routing overhead introduced. The milder increase in the routing overhead indicates that resizing the starting search area dynamically can reduce the number of control packets injected into the underlying network even under a high traffic loaded network.

Figure 5.8(b) shows the packet delivery ratios versus traffic loads for both protocols. It can be seen that the packet delivery ratios for both protocols decrease as the network load increases. As loads increase, the network will be more congested and this will lead to more packet loss, thus reducing the PDR values, although $TADL$ slightly outperforms LAR. The reason for the higher PDR by $TADL$ is because $TADL$ introduces less routing overhead as shown in Figure 5.8(a). In addition, $TADL$ also selects and uses the most trustworthy discovered path, and this can increase the chances of successful packet deliveries, which, in turn, could further reduce routing overheads. All these factors can reduce the chance of the network being congested. This is confirmed by the DLY results showed in Figure 5.8(c), which shows that LAR experiences much higher delays than $TADL$. Packets transmitted via a less congested network will experience fewer queuing delays at intermediate nodes thus less DLY. From these results, we can see that $TADL$ can cope with a higher level of traffic load.

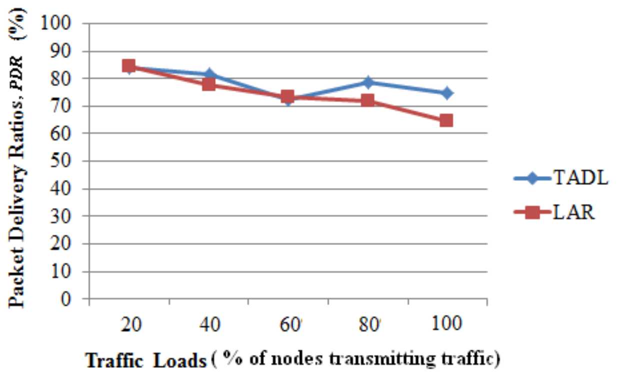
5.5.2.3 TADL vs. LAR: Varying Attacker Ratios

Figure 5.9 shows the effects of attacker ratios on the routing overheads, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$). For this set of simulation results, the traffic load used is 40% and the pause time is 600 s.

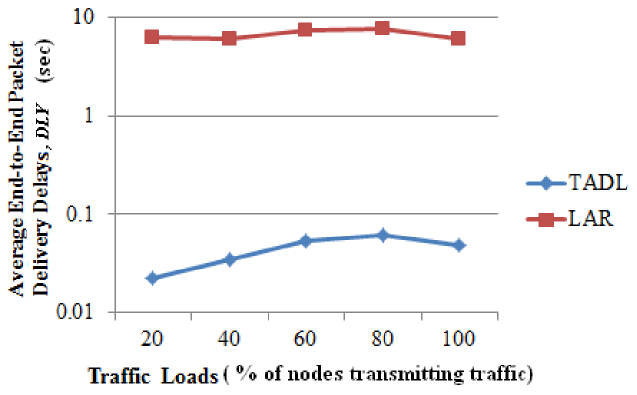
Figure 5.9(a) examines the effect of changing the attacker ratios on the routing overheads. From the figure, we can make two observations. The first observation



(a) Routing Overheads vs. Network Traffic Loads

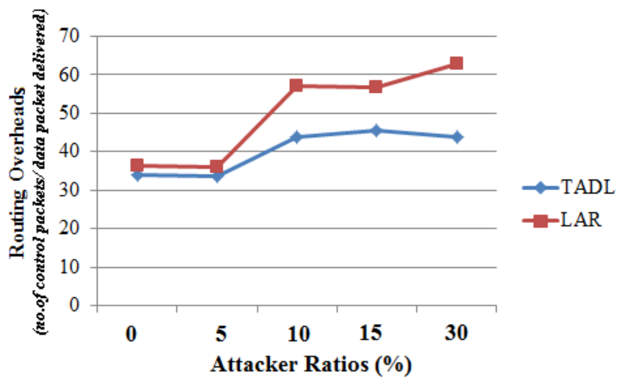


(b) Packet Delivery Ratios vs. Network Traffic Loads

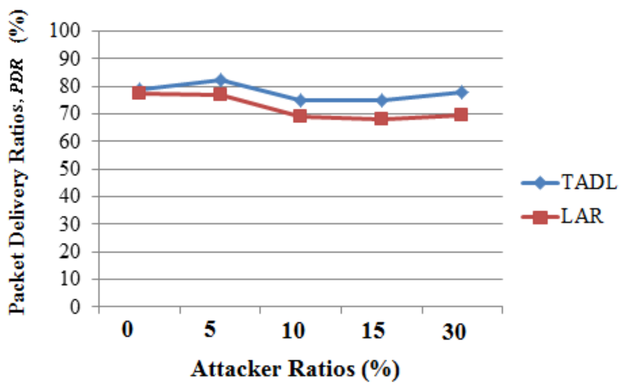


(c) Average End-to-End Packet Delivery Delays vs. Network Traffic Loads

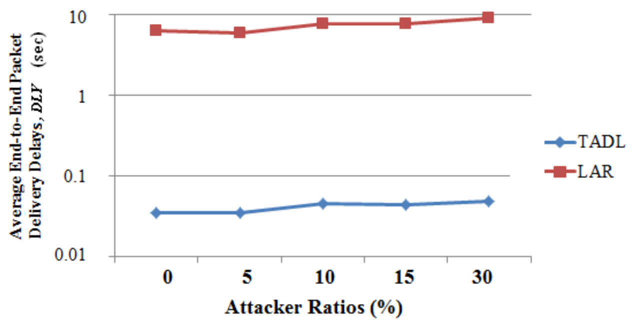
Figure 5.8: Routing Overheads, Packet Delivery Ratios and Average End-to-End Packet Delivery Delays vs. Traffic Loads for *TADL* and *LAR*



(a) Routing Overheads vs. Attacker Ratios



(b) Packet Delivery Ratios vs. Attacker Ratios



(c) Average End-to-End Packet Delivery Delays vs. Attacker Ratios

Figure 5.9: Routing Overheads, Packet Delivery Ratios and Average End-to-End Packet Delivery Delays vs. Attacker Ratios for *TADL* and *LAR*

is that for both protocols, an increase in the attacker ratio leads to an increase in the routing overhead in the network. This is because attackers drop data packets as well as control packets (in some cases). This can lead to restarting the path discovery process, if there is no alternative path available. Thus, more control packets are needed and more routing overheads introduced in the network. The second observation is that the routing overhead introduced by $TADL$ is lower than LAR in all cases of attacker ratios. When the attacker ratio in the network is low, the difference in the routing overhead is low. However, when the attacker ratio increases the difference increases to 41.4%, which means that when the attacker ratio reaches to 30% $TADL$ introduces 41.4% less routing overhead than LAR. This is because, unlike LAR, which selects the first discovered path to transmit the traffic, in $TADL$, path selection depends on the trust value of the path. If there is an attacker along a path, the trust value of the path will decrease, and the source node will avoid that path. Minimising the chance of data and control packet loss, thus the need for restarting the path discovery process, could reduce the routing overhead.

Similarly, Figure 5.9(b) that examines the effect of changing the attacker ratios on the packet delivery ratios, $PDRs$, shows that attacker ratios also have a negative impact on PDR, i.e. the higher the attacker ratio, the lower the PDR values. This is obvious, as packet dropping attacks will lead to packet loss, thus reducing $PDRs$. However, packet dropping attacks will also reduce network congestion, thus potentially offsetting the packet loss caused by congestion. From the figure, we can also see that the PDR of $TADL$ is better than that of LAR. This improvement is more significant when the attacker ratio is medium and high. For example, when the attacker ratio is 30%, the PDR of $TADL$ is 13% higher than that of LAR. This is due to two reasons. Firstly, the trust-based path selection approach used in $TADL$ helps to find more trustworthy paths, and more trustworthy paths can lead to increase PDR. Secondly, packet dropping attacks cause packet loss thus leading to PDR drops. At the same time, the attacks also reduce the traffic load in the network reducing the level of PDR drop caused by congestion. This can offset packet loss caused by network congestion.

Figure 5.9(c) shows the average end-to-end packet delivery delays, $DLYs$, versus attacker ratios. From the figure, we can make two observations. The first observation is that the DLY of $TADL$ significantly outperforms LAR. This is because $TADL$ discovers multiple path. So, when a path is untrusted, there are usually alternative paths that can be used immediately without the need of restarting a new path discovery process, and this can reduce the delay. In LAR, on the other hand, when a path is deselected or cannot be used, a new path discovery process is always involved and these results in longer delays and can make the network more congested. A congested network increases the queuing time, so the data packets take more time at the intermediate nodes to be served. The second observation is that when the attacker ratio increases, the DLY values for both protocols remain constant, i.e. do not change much. This can be explained as follows. When the attacker ratios increase, more packets will be dropped and this will trigger more path discovery processes leading to a higher level of routing overhead, as confirmed by the results shown in Figure 5.9(a). However, a higher attacker ratio also means less congestion, and the delays caused by network congestion will be reduced. In other words, packet dropping attackers actually offset the increase in DLY as caused by network congestion.

5.6 Chapter Summary

This chapter has presented the design and simulation study of a novel protocol, $TADL$, which aims to find multiple trustworthy paths for routing traffic towards a destination node while minimising overhead costs. $TADL$ can dynamically adjust the size of a starting search area in which route discovery packets are broadcast in response to the underlying network conditions, thus effectively reducing the routing overhead imposed on the network.

We use simulation to evaluate the effectiveness of the $TADL$ protocol in delivering traffic. The simulation study shows that $TADL$ provides a much better performance than LAR in terms of reducing routing overheads, increasing average end-to-end packet delivery delays and increasing packet delivery ratios. The study also shows that $TADL$ performs best in a stationary network or a network with a low to medium level of mobility. However, the dynamic adjustment measure used in $TADL$ is not effective in a highly mobile network. In the next chapter, we will present the design and evaluation of an improved version of the $TADL$ protocol, i.e. $TADLV2$ protocol, which is designed to overcome this weakness, making it effective in routing traffic under high network nodes' mobility level as well.

Chapter 6

TADLV2: An Improved Version of TADL

6.1 Chapter Introduction

This chapter presents the design and evaluation of the $TADLV2$ protocol. The $TADLV2$ protocol is an improved version of the $TADL$ protocol in Chapter 5. It overcomes the high routing overhead problem experienced by $TADL$ when the network mobility level is high. $TADLV2$ has used two ideas. The first is involving an intermediate node in the node-disjoint path discovery method when the neighbouring nodes' mobility level is high. The second idea is linking the starting search area resizing method to the node-disjoint path discovery methods applied to different neighbouring nodes' mobility level. These ideas attempt to make $TADLV2$ discovers paths with fewer control packets being injected into the underlying network, thus making the path discovery process more efficient under all mobility level scenarios.

This chapter is organised as follows. Section 6.2 gives the ideas used in the $TADLV2$ design. Section 6.3 describes the design of $TADLV2$ and its components. Section 6.4 describes the operations of the $TADLV2$ protocol. Section 6.5 presents the simulation study of the protocol. Finally, Section 6.6 concludes the chapter.

6.2 TADLV2: Ideas

This section describes the high-level ideas used in the design of the $TADLV2$ protocol. As mentioned in Chapter 5, $TADL$ was designed to discover trusted multiple path between a pair of source and destination nodes with minimum overhead costs. It has taken the following feature measures to reduce the routing overhead injected into the network. Firstly, it uses an adaptive directional approach to path discoveries. This can prevent unnecessary flooding of control packets across the entire network, but can also limit the number of paths that can be discovered. Secondly, to maintain a low bandwidth overhead cost, while, at the same time, discover a sufficient number of paths, $TADL$ adjusts the size of the starting search area dynamically. The size of the starting search area is adjusted in response to (a) the number of paths discovered in the previous path discovery phase, and (b) the underlying network conditions, namely neighbouring nodes' mobility level, neighbouring nodes' attacker ratios, and the time when the previous search was carried out. Simulation results have shown that $TADL$ outperforms LAR, the most relevant protocol, in a stationary network and a network with a low to medium level of mobility. However, when the network mobility level is high (e.g. when the network nodes move continuously, with 0 pause time), the performance of $TADL$ decreases significantly. This means that some of the design measures used in $TADL$ are not cost-effective or are even counterproductive when the mobility level is high.

To identify the problem, we have re-designed the $TADL$ protocol, producing $TADLV2$. The $TADLV2$ protocol uses two ideas. The first is to involve intermediate nodes in the multiple node-disjoint paths discovery at high neighbouring nodes' mobility level. The second is to link starting search area resizing to the multiple node-disjoint paths discovery algorithm. The two ideas are explained in the following subsections.

6.2.1 Idea 1: Intermediate Node Controlled Multiple Node-disjoint paths Discovery at High Mobility Level

This section describes the first idea used in $TADLV2$. The section first explains the existing work in the area of multiple path discovery processes. Then, it analyses the problem of the multiple path discovery used in $TADL$. It then explains the idea used in $TADLV2$ to rectify this problem.

In the existing work in the area of multi-path routing approaches, the multiple path discovery protocols may differ in terms of the types of paths they are designed to discover [78]. Basically, there are three types of paths, classified based on the level of disjointedness among nodes or links. They are node-disjoint paths, link-disjoint paths, and non-disjoint paths. Node-disjoint paths are paths that have no common nodes except the source and destination nodes. During a path discovery phase by such a protocol, the intermediate nodes reject all the duplicated copies of a *RREQ* packet. The Ad hoc On-demand Distance Vector Multi-path (*AODVM*) [126], the Geographic Multi-Path (*GMP*) routing protocol [68], the Node-Disjoint Multi-path (*NDMP-AODV*) routing protocol [54], and the Multi-path TCP Security (*MTS*) [62] are protocols designed to discover node-disjoint paths.

Link-disjoint paths are paths that have common nodes but no common links between a source and destination pair. During a path discovery phase by such a protocol, the intermediate nodes typically accept some of the duplicated copies of a *RREQ*, which have better attribute values than the first *RREQ* packet, e.g. lower hop counts. The accepted *RREQ* packets must come from different upstream nodes. The Split Multi-path Routing (*SMR*) protocol proposed in [58], and the Ad hoc On-demand Multi-path Distance Vector (*AOMDV*) routing protocol [72] are designed to discover link-disjoint paths.

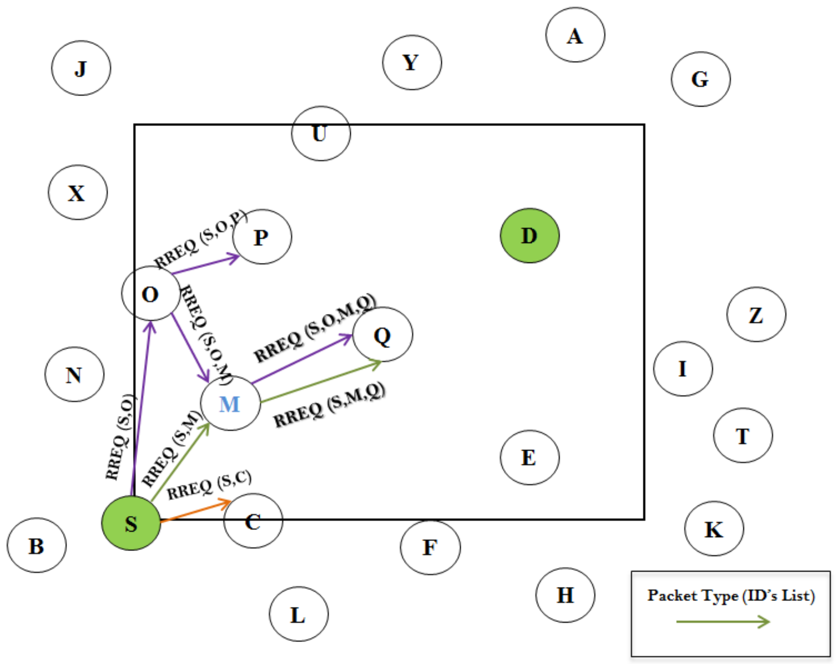
Non-disjoint paths are paths that share both nodes and links. In protocols designed to discover non-disjoint paths, intermediate nodes only play the role of packet forwarding. They are not involved in any decision making process; they simply accept and pass on all the *RREQ* packets they receive. Some of the non-disjoint path discovery protocols discover all the non-disjoint paths and then select a set of node-disjoint paths from the discovered paths. The Energy Aware Load Balancing Multi-path (*EALBM*) protocol in [24], the Trust-Aware Dynamic Location-Based (*TADL*) and the TOpology-HIding multi-path Protocol (*TOHIP*) in [130] are designed to discover non-disjoint paths.

There are trade-offs in discovering, selecting and using different types of paths, in terms of packet delivery reliability, routing overhead incurred in discovering them and the number of available paths that can be discovered. Using multiple node-disjoint paths has two advantages over the other types. Firstly, it offers the highest level of packet delivery reliability as if one intermediate node fails, only one path, i.e. the path containing the failed node, will be affected. Secondly, discovering node-disjoint paths imposes less routing overhead than link-disjoint and non-disjoint, as, during a path discovery phase, an intermediate node will reject all the duplicated copies of the same *RREQ* packet. However, the number of node-disjoint paths that can be discovered during a path discovery phase is also the least among the three path types. Conversely, protocols designed to discover multiple link-disjoint or non-disjoint paths do not reject duplicated *RREQ* packets at intermediate nodes. This allows the discovery of more paths. However, this also means that more *RREQ* packets will be injected into the underlying network resulting in more routing overheads.

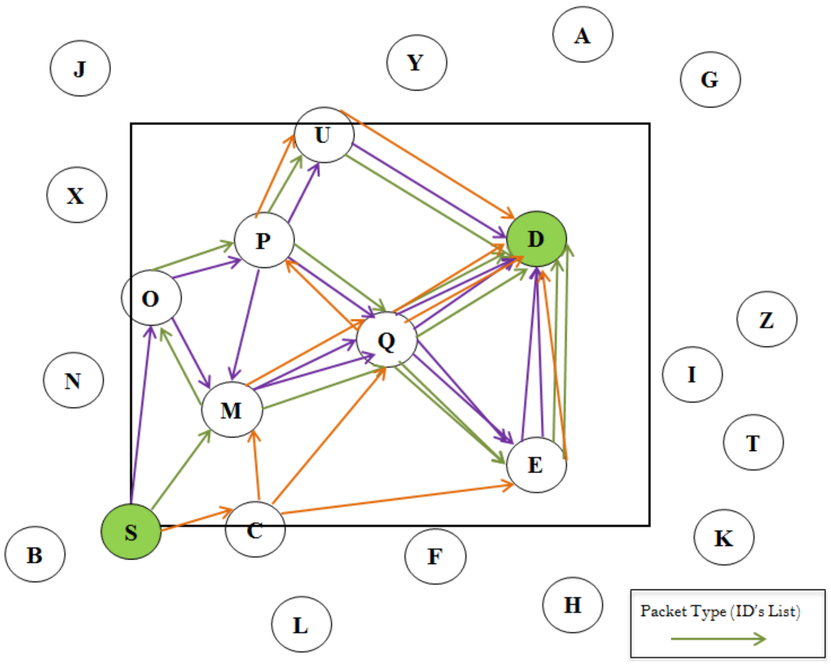
The $TADL$ protocol was designed as a node-disjoint path discovery protocol. The discovery of node-disjoint paths between a pair of source and destination nodes is carried out using the Source node Controlled node-disjoint Path Discovery *SC-MNPD* algorithm. The *SC-MNPD* algorithm is a two-step process. In step-1, the source node initiates a path discovery process by multicasting a path discovery packet in a specified search area. Once step-1 is complete, i.e. once all the paths in the search area are discovered, the source node executes step-2 to select a set of node-disjoint paths from the path discovered paths.

Figure 6.1 uses an example to illustrate the multiple path discovery process using the *SC-MNPD* algorithm. As shown in Figure 6.1(a), the source node, N_S, sends a *RREQ* packet to a selected set of neighbouring nodes, N_O, N_M and N_C. The set of nodes is selected based on the Trust-based Neighbouring Selection (TNS) algorithm, described in Section 5.3.2.3. When a node in this set, say N_M, receives the *RREQ* packet from N_S, it accepts the packet and passes it to a set of neighbouring nodes selected by N_M. If N_M receives another copy of the same *RREQ* coming from a different path, e.g. N_S->N_O->N_M, it also accepts the packet and passes it on. As shown in Figure 6.1(b), in this single path discovery phase, the destination node will eventually receive 14 copies of the same *RREQ* packet through different paths, and, for each *RREQ* it receives, the destination node will send a *RREP* back to the source node.

In a highly dynamic network, the paths discovered in step-1 and selected in step-2 would be highly likely to be obsolete due to the frequent topology changes. If these paths are used, packets transmitted over these paths are likely to experience a broken link and if this happens packets will be discarded and the upstream intermediate node of the broken link will return a *RRER* packet to the source node. The source node, upon the receipt of the *RRER* packet, will restart another path discovery phase if the source node cannot find an alternative path(s) in its routing table. So the higher the neighbouring nodes' mobility level, the more likely the links would break, the more *RRER* packets will be returned to the source node, and more route discovery phases will be initiated. As a result,



(a) Intermediate Node Accepts Second *RREQ*



(b) The Discovered Paths

Figure 6.1: *SC-MNPD* Algorithm

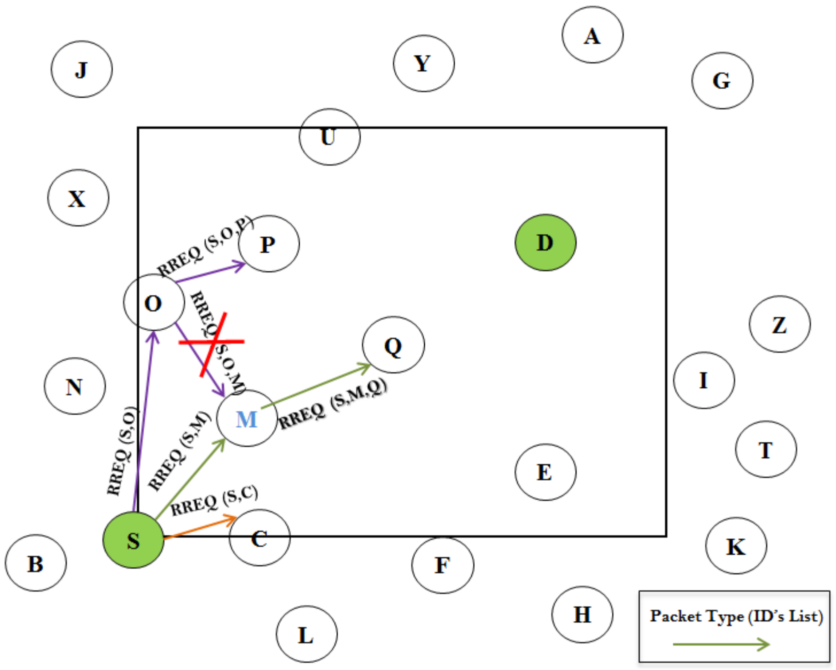
the more routing overhead will be generated. A higher level of routing overhead would be more likely to make the network more congested, causing more packets to be dropped.

To prevent the network getting into such a state, $TADLV2$ is introduced to further reduce routing overhead during multiple node-disjoint path discovery in a high neighbouring nodes' mobility level. The $TADLV2$ protocol uses a Mobility-based Adaptive Multiple Node-disjoint Path Discovery (*MA-MNPD*) algorithm. With this algorithm, $TADLV2$ uses the *SC-MNPD* algorithm (i.e. the algorithm described above and used in TADL) at a low or medium neighbouring nodes' mobility level, and switches to the Intermediate node Controlled Multiple Node-disjoint Path Discovery (*IC-MNPD*) algorithm if the neighbouring nodes' mobility level is high. With this *IC-MNPD* algorithm, it is not the source node, but every intermediate node, that filters out any duplicated $RREQ$ packets and selects node-disjoint paths during the process of packet forwarding. In this way, an intermediate node will only forward $RREQ$ packets along the selected node-disjoint paths. The transmission of the $RREQ$ packets along non-disjoint paths are prevented, thus reducing the number of control packet poured into the network.

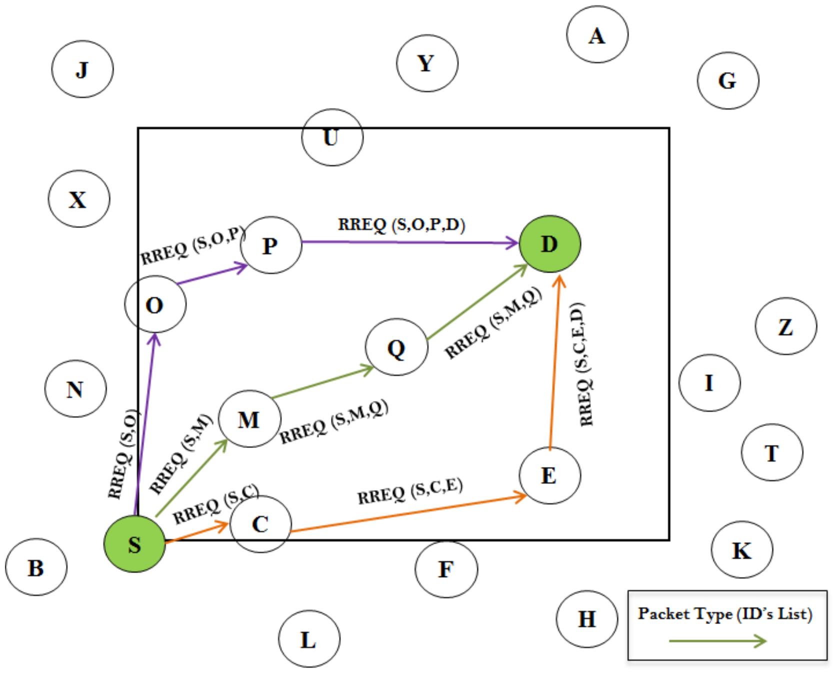
Figure 6.2 uses an example to illustrate the working of the *IC-MNPD* algorithm. As shown in Figure 6.2(a), the source node, N_S, sends a $RREQ$ packet to a selected set of downstream nodes, N_O, N_M and N_C. The set of nodes is selected based on the Trust-based Neighbouring Selection (TNS) algorithm, described in Section 5.3.2.3. Any node in this set filters out a duplicated $RREQ$. It receives and accepts the first $RREQ$ from N_S. Then, when a node in this set, say N_M, receives another copy of the same $RREQ$ coming from a different path, e.g. N_S->N_O->N_M, it will reject it. As shown in Figure 6.2(b), in this way, only one set of node-disjoint paths is discovered, but the number of control packets propagated in the network is reduced. The *MA-MNPD* algorithm is explained in detail in Section 6.3.

6.2.2 Idea 2: Linking Starting Searching Area Resizing to Multiple Node-disjoint paths Discovery Algorithm

This section describes the second idea used in $TADLV2$. It first analyses the problem of the starting search area resizing algorithm used in $TADL$, and then explains the idea that is used in $TADLV2$ to rectify this problem.



(a) Intermediate Node Rejects Second *RREQ*



(b) The Discovered Paths

Figure 6.2: $IC - MNPD$ Algorithm

Reducing the number of control packets in Idea 1 can be counterproductive to discovering a required number of paths between a pair of nodes. To resize the starting search area, $TADL$ uses the $A2SA$ algorithm. In the $A2SA$ algorithm, the size of the starting search area is adjusted dynamically in response to the underlying network conditions. Each path search can start from any of the three search area sizes (Area-S, Area-M or Area-L) based on the number of paths discovered in the previous discovery phase and the observations on the channel conditions of neighbouring nodes. When neighbouring nodes' mobility level is high, the discovery process starts with the smallest search area, Area-S, under the assumption that the discovered paths will likely to be valid. It then progressively moves to larger sizes if the number of paths discovered is fewer than the required.

$TADLV2$ uses a different approach to resizing the starting search area. It uses the same starting search area resizing algorithm as the one used in $TADL$, i.e. the $A2SA$ algorithm, in conjunction with the *SC-MNPD* algorithm, when the neighbouring nodes' mobility level is low or medium.

However, when the neighbouring nodes' mobility level is high, as $TADLV2$ uses the *IC-MNPD* algorithm, which employs the intermediate nodes to filter out any duplicated $RREQ$ packets, it makes more sense to use larger search area size to find more paths and to resist packet dropping attacks by intermediate nodes. In other words, $TADLV2$ switches to the Mobility-based Adaptive Adjusting Starting Searching Area (*MA-A2SA*) algorithm designed in conjunction with the *IC-MNPD* algorithm, when the neighbouring nodes' mobility level is high. With the *MA-A2SA* algorithm, when the neighbouring nodes' mobility level is high, $TADLV2$ starts the search from the largest search area, Area-L, which covers the whole network. The probability of finding a larger number of valid paths is higher with the use of a larger search area. With more paths discovered, the probability of having alternative paths when the paths are broken is higher. The *MA-A2SA* algorithm is explained in detail in Section 6.3.

6.3 TADLV2: Detailed Design

This section describes the $TADLV2$ protocol components and their design. Figure 6.3 shows the $TADLV2$ components and four of the components are $TADLV2$ specific. These four $TADLV2$ components along with the rest shown in the figure

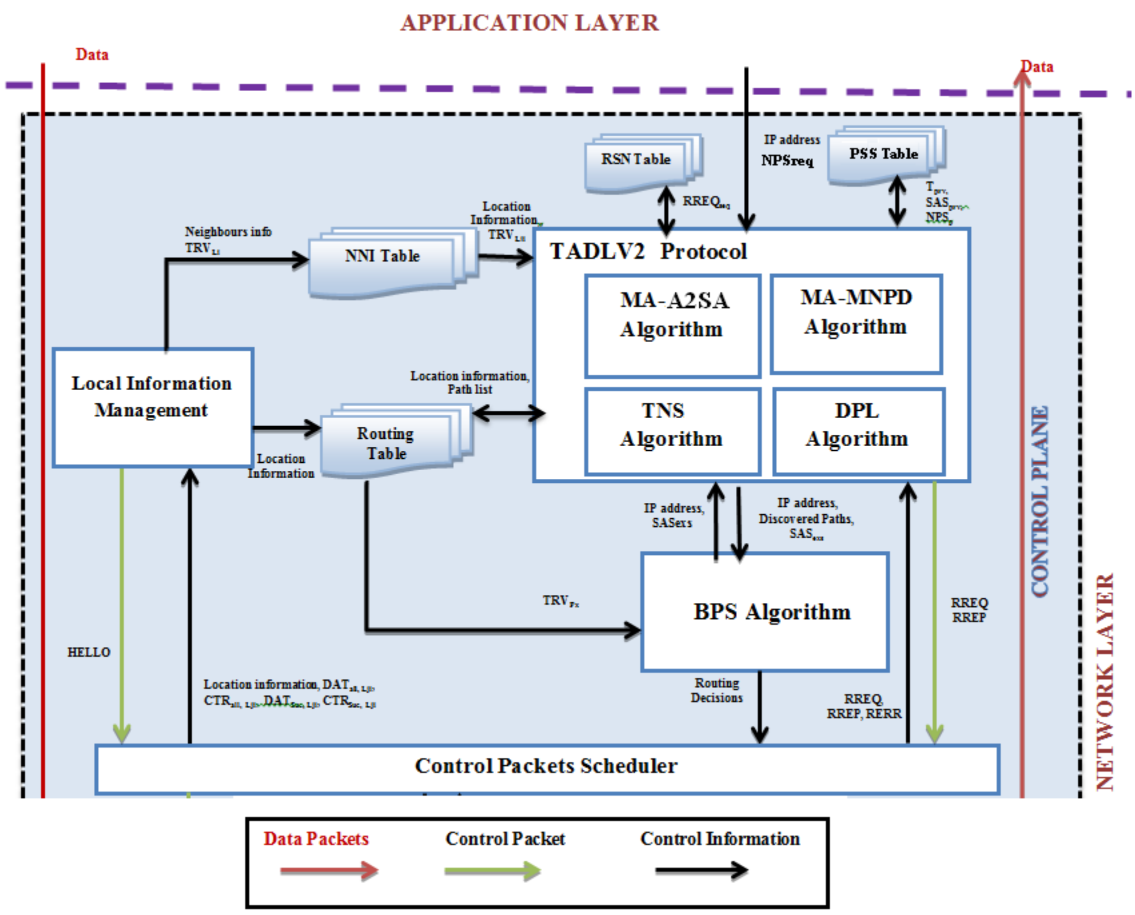


Figure 6.3: $TADLV2$ Components

collectively perform the functions defined for $TADLV2$. The $TADLV2$ components are the Destination Node Predicted Location (DPL) method, the Trust-based Neighbouring Selection (TNS) algorithm, the Mobility-based Adaptive Multiple Node-disjoint Path Discovery (MA-$MNPD$) algorithm and the Mobility-based Adaptive Adjusting Starting Searching Area (MA-$A2SA$) algorithm. The other components are the four tables, the Local Information Management component, and the Best Path Selection (BPS) algorithm. The tables and Local Information Management component have been described in detail in Chapter 3, the BPS, TNS, and DPL algorithm components are identical to those used in $TADL$ which have been described in detail in Chapter 5, so we have focused on describing the MA-$MNPD$ and MA-$A2SA$ algorithm components in details.

6.3.1 Mobility-Based Adaptive Multiple Node-disjoint Path Discovery (MA-MNPD) Algorithm

The *MA-MNPD* algorithm is used for discovering as many paths as possible between a pair of source and destination nodes, while, at the same time, reducing the routing overheads as possible under all network conditions, particularly when the network nodes are highly mobile. The mobility level is classified into two bands, low/medium mobility level and high mobility level. When the neighbouring nodes' mobility level is at a low/medium level, the *MA-MNPD* algorithm uses *SC-MNPD* algorithm (i.e. the algorithm used in $TADL$ Intermediate Node in Section 5.4.2), and when the neighbouring nodes' mobility level is high, *MA-MNPD* switches to the *IC-MNPD* algorithm. The mobility level of the neighbouring nodes is the average mobility of the neighbouring nodes. As described in Section 3.5.3.2, each node broadcasts its mobility speed to its one-hop neighbouring nodes in the $HELLO$ packets and the nodes maintain it in its NNI table.

The operations of the *SC-MNPD* algorithm are in two phases. In the first phase, the source node initiates a path discovery process in which all the non-disjoint paths linking the source and the destination nodes in the specified search area are discovered. In this phase, no effort is made to differentiate or identify path types (non-disjoint, link-disjoint, or node-disjoint paths). Once this process is complete, i.e. once all the available paths are discovered, the source node identifies and selects all possible sets of node-disjoint paths from the discovered paths [58]. This process is solely carried out by the source node, and the intermediate nodes are not involved. In other words, with this multiple node-disjoint paths discovery algorithm, all an intermediate node needs to do is to receive a $RREQ$ and forward it onto a set of selected neighbours. The set of neighbours is chosen based on their trust values (as described in Section 5.3.2.3). With this algorithm, duplicated $RREQ$ packets (i.e. multiple copies of the $RREQ$ packet with the same sequence number) are also forwarded.

The *IC-MNPD* algorithm, on the other hand, relies on intermediate nodes to select multiple node-disjoint paths while they are forwarding $RREQ$ packets. *IC-MNPD* is executed by intermediate nodes during a path discovery phase. An intermediate node will only pass a $RREQ$ packet once, it filters out any duplicated $RREQs$. Each node stores the sequence numbers carried in all the $RREQs$ that pass through the node in a $RREQ$ Sequence Number (RSN) table. As shown in Algorithm 6.1, upon the receipt of a $RREQ$, the intermediate node gets the $RREQ$ sequence number of the packet, $RREQ_{seq}$, and searches the RSN table for a match. If a match is found, the node will discard this $RREQ$ packet. Otherwise, the node will accept the $RREQ$ packet, and add (i.e. remember) the $RREQ$ sequence number, $RREQ_{seq}$, in the RSN table. At the end of the discovery process, only one set of node-disjoint paths will be discovered.

Algorithm 6.1 The *IC-MNPD* Algorithm

```
procedure IC-MNPD($RREQ$)
    Read $RREQ_{seq}$ from the $RREQ$
    Search $RSN$ table for $RREQ_{seq}$
    if $RREQ_{seq}$ is found then
        $RREQ$ Rejected
    else
        $RREQ$ Accepted
        Add $RREQ_{seq}$ to the $RSN$ table
        Add node's information to the path list in the $RREQ$ packet
        Read $IPaddressandSAS_{exs}$ from the $RREQ$ packet
        $predictedlocation \leftarrow$ DPL($IPaddress, SAS_{exs}$)
        $SelectedNeighbours \leftarrow$ TNS(predicted location)
        Send $RREQ$ to the $SelectedNeighbours$
    end if
end procedure
```

Upon accepting the $RREQ$ packet, the intermediate node adds the values of its attributes onto the path list in the $RREQ$ packet. These attributes are the node IP address and location information, and the trust values of the selected downstream neighbours. It then forwards the $RREQ$ packet on to the chosen neighbours that have been selected using the TNS algorithm (described in Section 5.3.2.3).

With the *MA-MNPD* algorithm, as shown in Algorithm 6.2, upon receiving a $RREQ$ packet, an intermediate node calculates the neighbouring nodes' mobility level, $AvrMob_{ngb}$. If the calculated value is high (above a predefined mobility level threshold, THR_{mob}), $TADLV2$ uses the *IC-MNPD* algorithm (the algorithm was used by the $TADL$ intermediate node as described in Section 5.4.2) for path discovery. Otherwise, if the calculated value is below this threshold value, it uses the *SC-MNPD* algorithm. All the threshold values used in our investigations are obtained experimentally (see Section 5.5.1).

Algorithm 6.2 The *MA-MNPD* Algorithm

```
procedure MA-MNPD($RREQ$)
    Read neighbouring nodes' $V_i$ from $NNI$ table
    Calculate the $AvrMob_{ngb}$
    if $AvrMob_{ngb} <= THR_{mob}$ then
        TADL-I($RREQ$)                                  ▷ SC-MNPD
    else
        IC-MNPD($RREQ$)                                 ▷ IC-MNPD
    end if
end procedure
```

6.3.2 Mobility-Based Adaptive Adjusting Starting Searching Area (MA-A2SA) Algorithm

The *MA-A2SA* algorithm is used to adjust the size of a starting search area used during a path discovery phase. As shown in Section 5.3.2.2, *MA-A2SA* implements a set of rules to govern the selection of the starting search area size for each route discovery process. There are three factors that should be considered when choosing a search area size to start a search. These factors are (1) neighbouring nodes' attacker ratios, (2) neighbouring nodes' mobility level, and (3) the time, T_{prv}, when the previous search attempt was carried out. The use of Factors (1) and (3) are identical to the ones from $TADL$.

Factor 2 - neighbouring nodes' mobility level: This is the average mobility level of the neighbouring nodes. If the neighbouring nodes' mobility level is high, *MA-MNPD* switches to the *IC-MNPD* algorithm. With the *IC-MNPD* algorithm, the intermediate nodes are responsible for filtering out any redundant *RREQ* packets to select a set of node-disjoint paths. In this case, it makes more sense to choose a larger search area, as this will allow us to find more paths and make the algorithm more resilient to any potential attacks and misbehaviour by the intermediate nodes. So, at the high mobility level, when using *MA-A2SA*, a source node starts the search from the largest search area, Area-L.

In detail, as shown in Algorithm 6.3, N_S starts by checking the attacker ratio of the neighbouring nodes. If it is above a predefined threshold value, THR_{att}, N_S starts the search from the largest search area, Area-L, which covers the whole network. Otherwise, N_S checks the neighbouring nodes' mobility level. If the neighbouring nodes' mobility level is above a predefined threshold value, THR_{mob}, N_S starts with the largest search area, Area-L.

Algorithm 6.3 The *MA-A2SA* Algorithm

```
function MA-A2SA($NPS_{req}$)
    if $AttackRatio >= THR_{att}$ then
        $SAS_{exs} \leftarrow Area - L$
    else
        Read neighbouring nodes' $V_i$ from $NNI$ table
        Calculate the $AvrMob_{ngb}$
        if $AvrMob_{ngb} >= THR_{mob}$ then
            $SAS_{exs} \leftarrow Area - L$
        else
            Read $SAS_{prv}$, $T_{prv}$, and $NPS_{prv}$ from $PSS$ table
            if $T_{cur} - T_{prv} >= THR_t$ then
                $SAS_{exs} \leftarrow Area - S$
            else
                if $NPS_{req} = NPS_{prv}$ then
                    $SAS_{exs} \leftarrow SAS_{prv}$;
                else
                    if $NPS_{req} < NPS_{prv}$ then
                        if $SAS_{prv} == Area - S$ then
                            $SAS_{exs} \leftarrow Area - S$;
                        else
                            $SAS_{exs} \leftarrow SAS_{prv} - 1$;
                        end if
                    else
                        if $SAS_{prv} == Area - L$ then
                            $SAS_{exs} \leftarrow Area - L$;
                        else
                            $SAS_{exs} \leftarrow SAS_{prv} + 1$;
                        end if
                    end if
                end if
            end if
        end if
    end if
    return $SAS_{exs}$
end function
```

Otherwise, if the neighbouring nodes' attacker ratio is less than THR_{att} and the neighbouring nodes' mobility level is less than THR_{mob}, the source node checks the time elapsed since the previous search T_{prv}, i.e. T_{cur}-T_{prv}. If T_{cur}-T_{prv} is above a specified threshold value, THR_t, the source node will start from the smallest search area, Area-S. Otherwise, the choice is made based on the number of discovered paths in the previous search, NPS_{prv}. If NPS_{req} is equal to NPS_{prv}, the existing search will start from the same search area size used in the previous search. If NPS_{req} is less than NPS_{prv}, then the existing search will use a smaller search area than the previous search provided that the previous search area used is not Area-S. If NPS_{req} is higher than NPS_{prv} the existing search will use a larger search area than the previous one, provided that the previous search area used is not Area-L.

6.4 TADLV2 Protocol Operations

This section describes the operations of the $TADLV2$ protocol in discovering multiple path. The operations of the protocol differ in terms of the node types in which it is executed. The operations are classified into $TADLV2$ source node operations, $TADLV2$ intermediate node operations, and $TADLV2$ destination node operations. As only the source and intermediate node components are modified in comparison with $TADL$, so this section only focuses on the discussion of the source and intermediate node operations designed for $TADLV2$. The destination node operations are identical to those used in $TADL$.

6.4.1 The TADLV2 Source Node Operations

When a source node, N_S, has traffic to send to a destination node, N_D, it first searches its routing table to see if there are already sufficient paths to N_D. If yes, N_S invokes BPS (explained in Section 5.3.2.4) that selects and uses the most trusted path from the paths in the routing table to transmit the traffic. If not, a path discovery phase is initiated by invoking the $TADLV2$ protocol as follows (Algorithm 6.4). N_S reads N_D's Location information from its routing table. If N_S does not find N_D's location information, N_S defaults to use the basic broadcast method to discover paths. In this case, the source node, N_S, and all the intermediate nodes that receive the $RREQ$ packet will broadcast the $RREQ$ packet to all their neighbouring nodes until the destination node is reached. If N_S can find N_D's location in its routing table, N_S invokes the *MA-A2SA* algorithm (as explained in Section 6.3.2) to determine a starting search area size. Then, N_S invokes the DPL algorithm (described in Section 5.3.2.1) to predict the current location of a destination node. The predicted location of the destination node is used to define N_D's search area. N_S then invokes the TNS algorithm (described in Section 5.3.2.3) to select the most trusted neighbours in the destination node search area. N_S then sends the *RREQ* packet only to these neighbours in this set.

Algorithm 6.4 The $TADLV2$ Source Node Operations

```
function TADLV2-S($IPaddress, SAS_{exs}, NPS_{req}$)
    Search the routing table for $N_D$'s location information)
    if D's location information is not found then
        $SAS_{exs} \leftarrow Area - L$
        $SelectedNeighbours \leftarrow Allneighbours$
    else
        if $SAS_{exs}$== Null then    ▷ First time $TADL$ is executed during this path discovery phase
            $SAS_{exs} \leftarrow$ MA-A2SA($NPS_{req}$)
        end if
        do
            $PredictedLocation \leftarrow$ DPL($IPaddress, SAS_{exs}$)
            $SelectedNeighbours \leftarrow$ TNS($PredictedLocation$)
            Send $RREQ$ to the $SelectedNeighbours$
            $N_S$ initiates a $RDTimer$ timer
            Wait for a $RREP(s)$ from $N_D$
            if $RREP$(s) packets arrived then
                Read $DiscoveredPaths$ from the $RREP$ packet
      return $DiscoveredPaths$
            else
                if $SAS_{exs}$ == Area-$L$ then
                    $N_D$ is an unreachable destination
                    $DiscoveredPaths \leftarrow Null$
      return $DiscoveredPaths$
                end if
                $SAS_{exs} \leftarrow SAS_{exs} + 1$
            end if
        while True
    end if
end function
```

Upon the transmission of a *RREQ* packet, N_S initiates a route discovery timer (*RDTimer*). By the expiry of this timeout interval, if N_S does not receive any *RREP* packets, a new *RREQ* will be transmitted in a larger search area than the one just used, i.e. $SAS_{exs} = SAS_{exs} + 1$, provided that the used search area, SAS_{exs}, is not the largest area, Area-L. If SAS_{exs} is equal to Area-L then N_D is considered an unreachable destination. If, by the expiry of *RDTimer*, N_S

receives *RREP* packet(s), N_S extracts the paths from the *RREP* packets. N_S then invokes *BPS* (explained in Section 5.3.2.4) to select the most trusted path from the discovered paths. N_S then sends the data traffic via the selected path to N_D.

6.4.2 The TADLV2 Intermediate Node Operations

During the path discovery phase, as shown in Algorithm 6.5, a $TADLV2$ intermediate node operates as follows. Upon receiving a *RREQ* packet, the intermediate node checks the path list in the *RREQ* packet. If the intermediate node IP address is listed in the path list in the *RREQ* packet, it means that this copy of the *RREQ* has already passed through this intermediate node before, the intermediate node discards the *RREQ* packet to prevent *RREQ* looping. Then the intermediate node invokes the *MA-MNPD* algorithm described in Section 6.3.2. In this algorithm, the intermediate node checks the neighbouring nodes' mobility level. If the neighbouring nodes' mobility level is high, the intermediate node rejects any duplicated *RREQ* packets and selects node-disjoint paths as it passes *RREQ* packets on. It accepts the *RREQ* only once. Otherwise, if the neighbouring nodes' mobility level is medium to low, the intermediate node accepts all the *RREQs* arrived. No effort is made to differentiate or identify path types. All the available paths are discovered, path selection is carried out at the source node at the end of the path discovery phase. It then forwards the *RREQ* packet on to the chosen neighbours. Intermediate nodes are responsible for forwarding *RREQ* packets towards the destination node until the destination node is reached. They do not reply to the *RREQ* packets.

Algorithm 6.5 The $TADLV2$ Intermediate Node Operations

```
procedure TADLV2-I(RREQ)
    Search RREQ path list for the node IP address
    if Node IP address is found in the RREQ path list then
        RREQ Rejected                                    ▷ Prevent Looping
    else
        MA-MNPD(RREQ)
    end if
end procedure
```

6.5 Simulation Study

This section investigates the performance of the $TADLV2$ protocol. The purpose of the investigation study is to check the effect of the ideas implemented in $TADLV2$ on reducing the level of routing overhead injected into the network when the mobility level is high. The investigation is carried out by comparing the performance of the $TADLV2$ protocol against that of the $TADL$ protocol (explained in Chapter 5) in terms of routing overheads, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$).

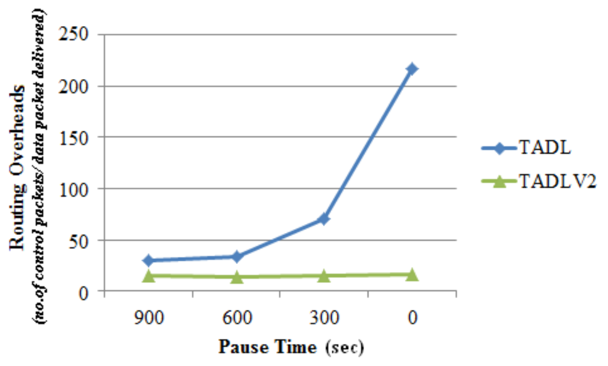
$TADLV2$ is designed to overcome the weakness of the $TADL$ observed from our investigation study. The weakness lies on how node-disjoint paths are discovered and determined when the mobility level is high. So, $TADLV2$ performs in exactly the same manner as $TADL$ when the neighbouring nodes' mobility level is low or medium; they only differ in how node-disjoint paths are discovered when the neighbouring nodes' mobility level is high. There are two foundational differences between the two protocols. The first difference is, in $TADL$, the discovery of the node-disjoint paths linking a source node and a destination node is done by using the *SC-MNPD* algorithm, whereas, $TADLV2$ uses the *MA-MNPD* algorithm that uses the *SC-MNPD* algorithm at a low or medium neighbouring nodes' mobility level, but switches to the *IC-MNPD* algorithm if the neighbouring nodes' mobility level is high. The *SC-MNPD* algorithm discovers and selects node-disjoint paths in two separate phases and relies on the source node to identify and select all possible sets of node-disjoint paths at the end of the path discovery phase. The *IC-MNPD* algorithm, on the other hand, discovers and selects node-disjoint paths in one phase, and relies on intermediate nodes to select and identify such paths during the process of packet forwarding. The intermediate nodes will filter out any duplicated $RREQ$ packets and only selects and forwards $RREQs$ that lead to the discovery of node-disjoint paths.

The second difference is that they use different algorithms to decide a starting search area size. $TADL$ uses the $A2SA$ algorithm. $A2SA$, when the mobility level is high, starts the discovery process with the smallest search area. The search area size progressively moves to larger sizes when the number of paths discovered is less than the required number of paths. On the other hand, $TADLV2$ uses the *MA-A2SA* algorithm. The *MA-A2SA* algorithm uses $A2SA$ when the neighbouring nodes' mobility level is at a low to medium level, but when the neighbouring nodes' mobility level is high, $TADLV2$ starts with the largest search area. These algorithms aim to reduce the number of control packets poured into the network under different mobility levels, but at the same time discover a sufficient number of paths.

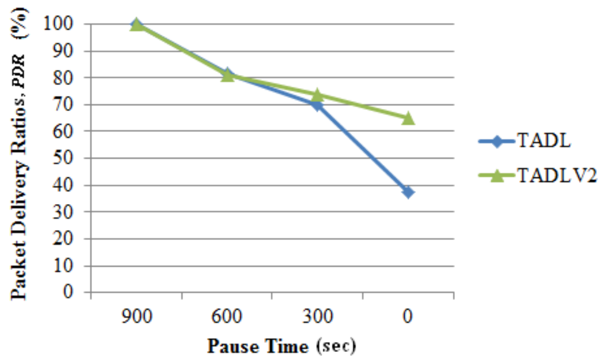
We study the effectiveness of these ideas by investigating the effects of varying network node mobility levels on the routing overheads, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$). Different pause times are used to reflect the network mobility levels. These are 0, 300, 600, and 900 s. The 0 s pause time means network nodes move continuously, i.e. the highest level of mobility. The use of 0 s to 300 s pause time is used to simulate a highly mobile network. The use of the 600 s pause time is used to simulate a medium level mobile network. For a stationary network, the pause time of 900 s is used. All other simulation parameter values used in the simulation study are identical to those specified in Section 4.6. The simulation results are presented in figures, where $TADL$ traffic and results are marked $TADL$ and the $TADLV2$ traffic and results are marked $TADLV2$ in Figure 6.4. Each result, i.e. each point in the figure is obtained by averaging 30 simulation runs and the confidence level is <= 2.86. In this set of results the traffic load used is that 40% of the nodes are source nodes and the attacker ratio is 0%.

From the figure, it can be seen that the $TADL$ and $TADLV2$ protocols perform similarly in a stationary network or a network with a low to medium level of mobility. However, when the network nodes' mobility level reaches the high level, $TADLV2$ significantly outperforms $TADL$. This means that the ideas implemented in $TADLV2$ are effective in accommodating a high level of mobility. Further discussions on the results are given in the following three subsections.

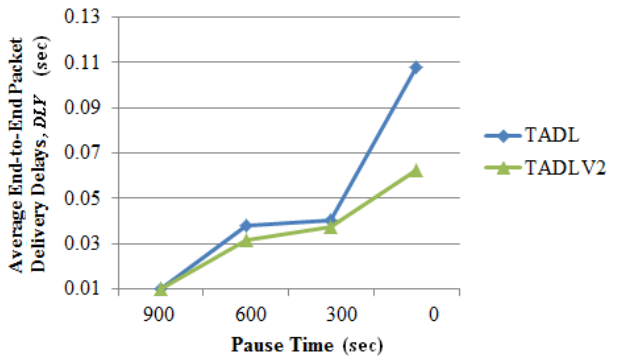
Figure 6.4(a) shows the effects of changing network nodes' mobility levels on the routing overheads. From the figure, we can see that when the network nodes' mobility level increases, in the case of $TADL$, the routing overhead increases. When the mobility level increases to 0 - 300 s pause time, the $TADL$ routing overhead increases sharply, this has been discussed in Section 5.5.2.1. For $TADLV2$, however, the increase in mobility level does not have much effect on the routing overhead. This is due to the reason that, unlike $TADL$ where all $RREQ$ packets will be passing the intermediate nodes, the higher the mobility level the more links break and more $RREQ$ will be handled by the intermediate nodes, $TADLV2$ does not require the intermediate nodes to process and forward all the $RREQ$ received, rather, an intermediate node only passes each $RREQ$



(a) Routing Overheads vs. Network Nodes' Mobility Levels



(b) Packet Delivery Ratios vs. Network Nodes' Mobility Levels



(c) Average End-to-End Packet Delivery Delays vs. Network Nodes' Mobility Levels

Figure 6.4: Routing Overheads, Packet Delivery Ratios, and Average End-to-End Packet Delivery Delays vs. Network Nodes' Mobility Levels for $TADLV2$ and $TADL$

once, filtering out any duplicated $RREQ$. Also as in $TADLV2$, the source node starts the search area from Area-L, this combining with the filtering out of duplicated $RREQ$ by intermediate node allows discovering of more node-disjoint paths without introducing excessive overhead in the network. From the figure, we can also observe that when the network nodes' mobility level is low to medium level, $TADLV2$ performs similarly as $TADL$. This is because both protocols use the same methods and algorithms in these cases.

The routing overhead reduction in $TADLV2$ indicates that to discover multiple node-disjoint paths in a high mobility network, using a large search area size, at the same time filtering out duplicated $RREQ$ packets is an effective approach.

Figure 6.4(b) examines the effect of changing network nodes' mobility levels on the packet delivery ratios. From the figure, we can make two observations. The first observation is that the PDR decreases as the network mobility level increases (explained in Section 5.5.2.1). The second observation is that the PDR of $TADLV2$ is better than $TADL$ in a high mobility network. This is because $TADLV2$ produces less routing overhead than $TADL$ in the case of high network nodes' mobility levels. Lower routing overhead decreases the chance of network congestion, thus allowing more packets to be delivered.

Figure 6.4(c) shows the effect of changing network nodes' mobility levels on the average end-to-end packet delivery delays. It can be seen that the average end-to-end packet delivery delays increase as the network mobility level increases (this has been explained in Section 5.5.2.1). The figure also shows that $TADLV2$ performs better than $TADL$ in a high mobility network with 57.3% reduction in delays at 0 s pause time. This is a consequence of the reduced routing overhead of $TADLV2$ as shown from the results of Figure 6.4(a), whereas, in $TADL$, the high routing overhead congests the network, which increases the data packet queueing time at the congested intermediate nodes.

6.6 Chapter Summary

This chapter has presented the design and simulation study of a novel multiple node-disjoint paths discovery protocol, $TADLV2$. $TADLV2$ is designed to overcome the weaknesses identified in the $TADL$ protocol, using different algorithms to discover node-disjoint paths based on the neighbouring nodes' mobility levels. It uses a source node controlled node-disjoint path discovery algorithm, i.e. *SC-MNPD*, when the mobility level is low or medium, but an intermediate node controlled node-disjoint path discovery algorithm, i.e. *IC-MNPD*, when the mobility level is high. This approach, combined with the use of a mobility-based adaptive starting search area adjustment algorithm, i.e. *MA-A2SA*, enables multiple node-disjoint paths to be discovered with lower routing overhead in all mobility level situations. The simulation results have shown that this approach provides a much better performance than the approach used in $TADL$.

There are three issues that we can improve in $TADLV2$. The first is that there are pros and cons with the use of the two different multiple node-disjoint path discovery approaches, *SC-MNPD* and *IC-MNPD*. The former places less trust on intermediate nodes, but introduces more routing overhead into the underlying network, whereas the latter assumes more trust on the intermediate nodes, but can reduce routing overhead by 76.4% at 0 s pause time. The second issue is that $TADLV2$ searches for multiple path, but it only uses one path to deliver the traffic. It is not always possible to find a single path that is trustworthy and has sufficient bandwidth to deliver high priority traffic. Thus, there is clearly a need for making use of all the available bandwidth to provide better performance in supporting QoS. The third issue is that $TADLV2$ treats all the traffic in the same way. It does not differentiate the delivery of high and low priority traffic. The work presented in the next chapter addresses these issues, investigating how to harvest the bandwidth of multiple path to speed up the delivery of high priority traffic, to better supporting QoS, and how to deal with unreliability by intermediate nodes in the process of traffic delivery. It also investigates how to identify and counter any misbehaviour by the intermediate nodes during the path discovery phase to lower the level of trust on the intermediate nodes.

Chapter 7

SAV: Path Selection, Traffic Allocation and Path Verification to Support High Priority Traffic Delivery

7.1 Chapter Introduction

This chapter presents the design and evaluation of the path Selection, traffic Allocation, and path Verification (SAV) solution. SAV offers three major functions: (1) it can deliver high priority traffic over multiple path; (2) it uses packet duplication for paths that are not sufficiently reliable; (3) it allows a source node to verify the delivery reliability based on the feedback by the destination node. These functions are used to make the best use of the available bandwidth, and to increase the delivery ratio of high priority traffic.

The rest of this chapter is organised as follows. Section 7.2 reviews and critically analyses related works in the area of multi-path delivery. Section 7.3 describes the ideas used in the SAV solution design. Section 7.4 describes the SAV design and its components. Section 7.5 describes the SAV operations. Section 7.6 discusses the simulation study of the solution. Finally, section 7.7 concludes the chapter.

7.2 Related Works

Discovering multiple path for multi-path delivery of traffic from a source node to a destination node involves three processes: multiple path discovery, path selection, and traffic allocation over the selected paths.

The Path Discovery Process: This process handles the discovery of multiple path in a single discovery process. A multiple path discovery process differs in terms of the multiple path discovery protocol used, which is defined to discover some specific types of paths. The multiple path discovery process has been discussed in detail in Chapter 6.

The Path Selection Process: Once multiple path are discovered, one path (or a set of paths) from the discovered set of paths should be selected and used to deliver the traffic. The selection of the path(s) is dependent on the purpose of the discovery protocol described. The protocol can be delay-aware, reliability-aware, or disjointedness-aware protocols. For the delay-aware protocols, the paths are selected such that the average delays in delivering traffic should be as low as possible. Examples of the delay-aware protocols are the Redundant Multi-path Source Routing (*RMSR*) protocol [127], the Delay-Aware *AODV* Multi-path (*DAAM*) protocol [17], and Multi-path TCP Security (*MTS*) protocol [62].

Reliability-aware multiple path discovery protocols are typically designed to prolong the network lifetime, e.g. spreading the load across the network nodes, minimising the chance of some part of the network being overloaded. They select paths consisting of the most reliable set of nodes, in terms of battery life and/or available bandwidth. The Congestion Adaptive Multi-path Routing protocol for Load Balancing (*CAMRLB*) [4], the Multi-Path Dynamic Source Routing (*MP-DSR*) protocol [60], the Energy-aware Multi-path Routing Protocol (*EMRP*) [61], and the Energy Aware Load Balancing Multi-path (*EALBM*) routing protocol [24] are examples of this protocol type.

Disjoint multiple path discovery protocols select paths that are node-disjoint paths to enhance traffic delivery reliability. The 2-Dimensional Adaptation ARChitecture (*2-DAARC*) [75], the Multi-Path *AODV* (*MP-AODV*) routing protocol [26], and the Valid Source-Destination Edges (*VSDE*) protocol [47] are examples of this protocol type.

The traffic allocation over the selected paths process: This process deals with how the data is distributed among the selected paths. The traffic allocation strategies used in multiple path discovery protocols can be classified into four categories, backup path(s), parallel paths, duplicated traffic, and hybrid strategies.

The backup path(s) strategy is used in most multiple path discovery protocols.

With this strategy, one of the discovered paths is used as a primary path, and the others are used as secondary or backup paths. If the primary path breaks, packets can be promptly delivered along one of the secondary paths. Examples of protocols using this strategy are the Improved Trust-Aware Dynamic Location-based ($TADLV2$) Protocol [11], the Ad hoc On-demand Multi-path Distance Vector protocol ($AOMDV$) [31, 73], Multi-path Routing Protocol (MRP) [30], and $AODV$ Backup Routing (*AODV-BR*) protocol [57]. However, to best support delivery of high priority traffic, it may be necessary to use more than one or all the available paths.

With the parallel paths strategy, a set of discovered paths is used in parallel, each supporting the delivery of a fraction of the traffic to be delivered. Using multiple path in parallel can provide load balancing and reduce congestion risks in MANETs. It can also harvest more bandwidth to speed up traffic deliveries. Examples of this strategy are the Disjointed Multi-Path routing based on the Optimized Link State Routing (*DMP-EOLSR*) [41], the *QoS* and Load Balancing $AOMDV$ (*QLB-AOMDV*) [113, 23], the Energy Aware Load Balancing Multi-path ($EALBM$) [24], and the Multi-path source routing (MSR) [119] protocols.

With the duplicated traffic delivery strategy, one of the discovered paths is used as a primary path to send traffic, and a secondary or backup path(s) is used to duplicate the traffic delivery. Duplication is used to increase delivery reliability, i.e. to increase the chances of data packets being successfully delivered to their destination. Both the Redundant Multi-path Source Routing ($RMSR$) [127] and the Redundant Source Routing (RSR) [120] protocols use this approach.

The hybrid strategy switches between different strategies depending on the network condition. The Congestion Adaptive Multi-path Routing protocol for Load Balancing ($CAMRLB$) [4] and the 2-Dimensional Adaptation ARChitecture (*2-DAARC*) [75] protocols use this strategy.

7.3 SAV: Ideas

SAV is designed to optimise high priority traffic delivery by using multi-path deliveries, packet duplication over the multiple path in adaptation to bandwidth requirement of traffic, available bandwidth, trust value of each path, and destination node feedback to verify the delivery reliability. As mentioned in Chapter 1, there are several data types in the network. Some of these data types have a higher priority than others. To support the QoS requirements of the high priority data types effectively, it is necessary to satisfy the minimum user's level or application level bandwidth requirements. In the $TADLV2$ protocol design (described in Chapter 6), we use path trust values to govern the selection of the most trustworthy path from the discovered paths to route the traffic. However, due to node mobility and packet dropping attackers, available bandwidth is fluctuating, and it is not always possible to find a single path that is both trustworthy and has the required bandwidth to support the delivery of high priority traffic. Therefore, there is need for making use of the available bandwidth provided by all the discovered paths. The question is how to harvest the fluctuating available bandwidth from multiple path to serve the delivery of high priority traffic and do so with as little overhead as possible.

The SAV solution can also resist any misbehaviour by an intermediate node during the path discovery phase. During the path discovery phase, the $TADLV2$ protocol uses two different multiple node-disjoint path discovery approaches, *SC-MNPD* and *IC-MNPD* to reduce the overhead costs. There are pros and cons with the use of the two different multiple node-disjoint path discovery approaches. *SC-MNPD* places less trust on intermediate nodes, but introduces more routing overhead into the underlying network, whereas, *IC-MNPD* assumes more trust on the intermediate nodes, but can reduce routing overhead. In addition, during the data transmission phase, the path available bandwidth, estimated during the path discovery process, may not be accurate for the whole phase due to node mobility and packet dropping attacks. Therefore, there is a need for the validation of the available bandwidth and trust values of a path.

SAV has two design criteria: (1) if no single path can be found which satisfies the bandwidth requirement of the high priority traffic, traffic should be spread over multiple path, including those with low trust values; (2) during the data transmission phase, the source node should be able to adjust the path quality (path available bandwidth and trust values) estimated during the path discovery phase based on feedback from the destination node. These criteria are taken to make the best use of the available bandwidth, and to optimise the reliability of the high priority traffic deliveries.

To meet these design criteria, four measures have been used in the SAV design. The first is to differentiate traffic types (low priority and high priority) and deliver each traffic type with a different delivery strategy. Low priority traffic delivery uses a Single Path Service (SPS) and this is done by selecting the path with the highest trusted value from those discovered (this is the case used in the $TADLV2$ protocol design). For low priority traffic delivery, no effort is made by the source node to check the bandwidth requirement of the traffic. For high priority traffic delivery, on the other hand, two services are used to accommodate bandwidth requirements versus the estimated available bandwidths and trust values along the paths discovered. These two services are the SPS and Multiple Path Service (MPS). If the available bandwidth of a single path that has been discovered can satisfy the bandwidth requirement of high priority traffic, then SPS is used. Otherwise, if the source node cannot find a single path with sufficient bandwidth and trust value, MPS will be used. With MPS, the most trustworthy set of node-disjoint paths, which together could satisfy the bandwidth requirement of a high priority traffic, is selected and used. This differential treatment of traffic is explained in detail in the QoS Path Selection (QPS) algorithm component in Section 7.4.1.

The second measure is to allocate the high priority traffic on the selected paths according to the paths' available bandwidths estimated during the path discovery phase. A path with a higher available bandwidth will be assigned more traffic.

The third measure is to duplicate the traffic on the paths with low trust values. In this way, if there is a shortage of trustworthy paths, we can make use of paths that are not trusted but offset the probability of packet loss by duplicating transmissions over these paths. The second and third measures are explained in the QoS Traffic Allocation (QTA) algorithm component in Section 7.4.2.

The three measures discussed above are to optimise the use of available bandwidth for high priority traffic delivery. While achieving this, these measures can also help with load balancing making the solution more resilient to traffic congestion and packet dropping attacks.

The fourth measure is to allow the source node to validate the estimated bandwidth and trust values of each used path based on the feedback by the destination node and adjust the traffic allocation over the paths accordingly. The feedback indicates the number of data packets received in a given slot during the data transmission phase. The source node will compare the number contained in the feedback with the number of data packets sent. Based on this comparison, the source node will validate and adjust the path's available bandwidth and trust values discovered during the path formation phase. In this way, the intermediate nodes may lie about a link bandwidth capacity, but this can be detected once the destination feedback is received. Once a path's available bandwidth and trust values are verified and updated, the source node will reselect a path or a set of paths based on the updated values and adjust the allocation of the high priority traffic on the selected paths. This measure is explained in detail in two algorithms. They are the Path Quality Adjusting (PQA) algorithm and the DFB Packet Creation (DPC) algorithm, which are explained in Section 7.4.3 and Section 7.4.4, respectively.

7.4 SAV: Detailed Design

This section describes the SAV solution design in detail. Figure 7.1 shows the SAV components. Two of the SAV solution components, the QoS Paths Selection (QPS) algorithm and the QoS Traffic Allocation (QTA) algorithm are invoked during the path discovery phase. The other two SAV solution components, the Path Quality Adjusting (PQA) algorithm and the DFB Packet Creation (DPC) algorithm components are invoked during the data transmission phase. These four SAV components along with the rest shown in the figure collectively perform the functions defined for SAV. The rest of the components shown in the figure are the five tables, the Local Information Management component, and the $TADLV2$ protocol component. The tables and the Local Information Management component have already been described in Chapter 3. The $TADLV2$ protocol has been described in Chapter 6. In this section, we focus on describing the QPS, QTA, PQA, and DFB algorithms.

It should be pointed out that in the description of the $A2SA$ and $MA\text{-}A2SA$ algorithm components used in $TADLV2$ in Chapter 6, the number of required paths, NPS_{req}, is used. However, in this chapter, the bandwidth requirement (BW_{req}) of an application is used, in the $MA\text{-}A2SA$ algorithm, instead. BW_{req}, measured in packets/s, is determined based on the QoS requirements of the traffic.

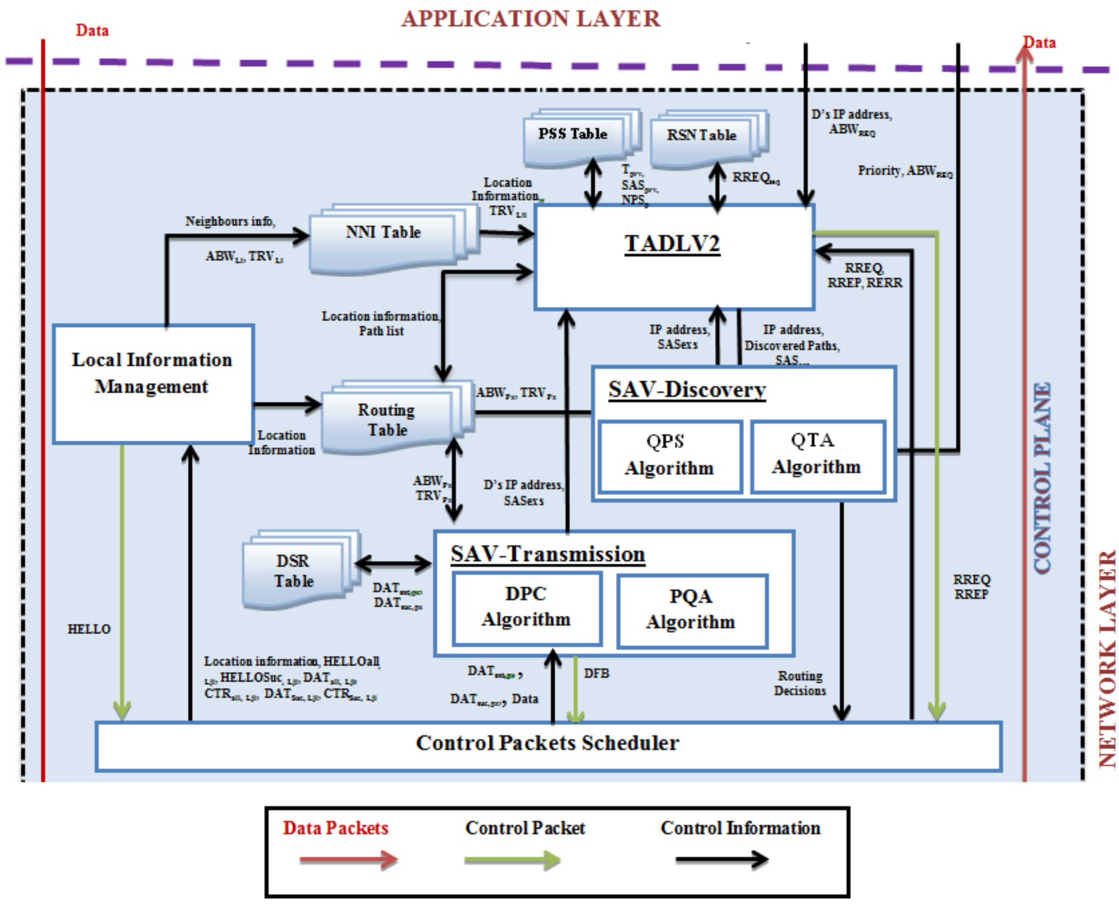


Figure 7.1: *SAV* Components

7.4.1 QoS Path Selection (QPS) Algorithm

The QPS algorithm selects one path or a set of paths from the paths discovered so far. The selection is based on the traffic priority and the paths quality (available bandwidths and trust values). This path selection process can be described using the following five steps:

Step 1. Assessing Path Availability and Calculating Path Quality: This step assesses the availability of paths and calculates the available bandwidths and trust values of the discovered paths. The path is available if the source node receives a $RREP$ from the destination node that contains the path list of all intermediate nodes forming the path. The source node checks the $RREPs$ arrived from the destination node. If no $RREP$ arrives, the source node, N_S, will restart the path discovery process. In the new path discovery process, the source node will use a larger search area than the one just used. Otherwise, if N_S receives $RREP$

packets, it calculates the path quality of each path discovered. The path quality measures both available bandwidth, ABW_{P_x}, and trust value, TRV_{P_x}, of the path. These values are estimated based on the weakest link principle. The path available bandwidth, ABW_{P_x}, is the minimum available bandwidth of the links forming the path, P_x. This is also called the bottleneck bandwidth of the path. Similarly, the path trust value, TRV_{P_x}, is the minimum trust value of the links forming P_x.

Step 2. Classifying Paths: In this step, the source node classifies the paths into two groups (types) based on their trust values versus a threshold value. One group contains the trusted paths and the other group contains the untrusted paths. A path is considered as trusted if the trust value of the path, TRV_{P_i}, is equal to or greater than a threshold, THR_{TRV}. A path is classified as untrusted, if the trust value of the path, TRV_{P_i}, is lower than THR_{TRV}.

Step 3. Choosing a Delivery Service: In this step, the source node determines a delivery service (SPS or MPS) that will be used to deliver the traffic. If the traffic is low priority, the most trusted path will be used. Otherwise, if the traffic is high priority, the source node ranks the trusted paths based on their available bandwidths. It then checks the bandwidth required by the traffic. If a single trusted path, that alone could satisfy the required bandwidth, BW_{req}, has been found, then the Single Path Service (SPS) is used. If more than one trusted path that satisfies the required bandwidth has been discovered, then the one with the highest trust value will be used. Otherwise, MPS is chosen.

Step 4. Discovering Path Sets: If MPS is chosen, the source node will proceed to find all the possible sets of node-disjoint paths. The paths in each set should collectively satisfy the bandwidth required by the traffic. This step is performed using Algorithm 7.1.

Step 5. Selecting a Path Set for Traffic Delivery: In this step, the source node selects a set of paths to deliver the high priority traffic. The source node calculates the available bandwidth of each set discovered in Step (4). The available bandwidth of a set is the sum of the available bandwidths of the paths in that set. If more than one set is found, the source node will calculate the trust value of each set. The trust value of a set is the minimum trust value of the paths in that set. The source node then ranks the sets that satisfy the required bandwidth based on their trust values and selects the set with the highest trust value. Otherwise, if no set can satisfy the required bandwidth, then the source node will

Algorithm 7.1 The *SNP* Algorithm

```
function SNP($DiscoveredPaths$)
    $x \leftarrow 0$                                                ▷ To count the number of sets
    for k=0; k < number of $DiscoveredPaths$; k++ do
        for Every path $P_i$ in $DiscoveredPaths$ do
            $y \leftarrow 0$                              ▷ To count the number of paths in each set
            $PathSet(x)(y) \leftarrow P_i$
            for Every path $P_j$ in the discovered paths $DiscoveredPaths$ do
                if $P_j$ is node-disjoint path with all the paths in the set then
                    if This set will not be identical to any set in $PathSet$ then
                        $PathSet(x)(y) \leftarrow P_j$
                        $y \leftarrow y + 1$                         ▷ Next path in this set
                    end if
                end if
            end for
            $x \leftarrow x + 1$                                  ▷ Next set in the set of paths
        end for
    end for
     return $PathSet$
end function
```

restart a route discovery process with a bigger search area. If the previously used search area is already the largest area, Area-L, then N_D is considered to be an unreachable destination. The five steps are summarised in Algorithm 7.2.

Algorithm 7.2 The *QPS* Algorithm

```
function QPS($DiscoveredPaths, BW_{req}, IPaddress, SAS_{exs}$)
    if $DiscoveredPaths$ == Null then
        if $SAS_{exs}$ == Area-$L$ then
            $N_D$ is an unreachable destination
     return Null
        else
            $SAS_{exs} \leftarrow SAS_{exs} + 1$
            $DiscoveredPaths \leftarrow$ TADLV2-S($IPaddress, SAS_{exs}$)
        end if
    else
        for Every x path in $DiscoveredPaths$ do
            Calculate the $ABW_{Px}$
            Calculate the $TRV_{Px}$
        end for
        $SelectedPath \leftarrow Null$                                    ▷ Step3
        Rank $DiscoveredPaths$ based on $ABW_{Px}$ then $TRV_{Px}$
```

Algorithm 7.2 The QPS Algorithm (continued)

```
        if $ABW_{P1} >= BW_{req}$ then
            if $TRV_{P1} >= THR_{TRV}$ then
                $SelectedPath \leftarrow P_1$
            else
                $DiscoveredSets \leftarrow$ SNP(DiscoveredPaths)                ▷ Step4
                for Every i set in $DiscoveredSets$ do                          ▷ Step5
                    Calculate the set available bandwidth $ABW_{Si}$
                    Calculate the set trust value $TRV_{Si}$
                    if $ABW_{Si} <= BW_{req}$ then
                        Remove $Si$ from $DiscoveredSets$
                    end if
                end for
                if $DiscoveredSets ==$ Null then
                    if $SAS_{exs} ==$ Area-$L$ then
                        $N_D$ is an unreachable
      return Null
                    else
                        $SAS_{exs} \leftarrow SAS_{exs} + 1$
                        $DiscoveredPaths \leftarrow$ TADLV2(SAS)
                    end if
                else
                    Rank $DiscoveredSets$ based on the sets trust values
                    $SelectedPath \leftarrow DiscoveredSets_1$
                end if
            end if
        end if
    end if
      return SelectedPath
  end function
```

7.4.2 QoS Traffic Allocation (QTA) Algorithm

The QTA algorithm is used to assign the high priority traffic to a selected path or a set of selected paths. The paths in the set are all node-disjoint paths and they may have different bandwidths and trust values. Packets sent over the paths with trust values below a given threshold will be duplicated to enhance delivery reliability.

Once a path or a set of paths are selected by the QPS algorithm, QTA is invoked to assign the high priority traffic to the selected paths as shown in Algorithm 7.3. The number of packets allocated to each path is based on the path type (explained in Section 7.4.1) and the available bandwidth offered by the path. In detail, the following rules apply:

- For a trusted path (i.e. path with its $TRV_{P_x} >= THR_{TRV}$), the number of packets allocated to the path is dependent on the ABW_{P_x} value of the path, and in this case, no packet duplication is applied.
- For an untrusted path (i.e. path with its $TRV_{P_x} < THR_{TRV}$), the number of packets allocated to the path is also based on its ABW_{P_x} value. In addition, each packet transmitted on this path is duplicated, i.e. another copy of the same packet is constructed and allocated to a different path based on the ABW_{P_x} value of the path. For every packet duplication, the source node slows a low priority packet transmission. This measure is taken to prevent network congestion while applying data duplication. In other words, the packet duplication slows down low priority packet injection into the network, but in the same time, it increases the delivery reliability of high priority traffic.

Algorithm 7.3 The *QTA* Algorithm

```
procedure QTA($SelectedPaths, data$)
    for Every traffic do
        for x=0; Px on $SelectedPaths$; ++x do
            if $TRV_{Px} >= THR_{TRV}$ then
                Allocate data to $P_x$ based on $ABW_{P_x}$
            else
                Allocate data to $P_x$ based on $ABW_{P_x}$
                Duplicate data
                $x \leftarrow x + 1$                              ▷ Use next path
                Allocate duplicated data to $P_x$ based on $ABW_{P_x}$
            end if
        end for
    end for
end procedure
```

7.4.3 Path Quality Adjusting (PQA) Algorithm

The *PQA* algorithm counts data packets sent from the source node via each selected path and adjusts the path quality values during the data transmission phase. Upon the transmission of the first data packet, the source node, N_S, initiates a DFB timer ($DFBwait$). During this period, i.e. before the expiration of the $DFBwait$, N_S counts the data packets sent to the destination node, N_D, over this path, P_x, and records the counter value in the $DAT^{SD,T}_{snt,Px}$ field in the DPC table. N_S then waits for the arrival of a DFB packet from the destination node. This DFB packet will contain the number of data packets, $DAT^{SD,T}_{suc,Px}$, successfully received by N_D.

When N_S receives the DFB packet, it compares the $DAT^{SD,T}_{suc,Px}$ value carried in the DFB header with the $DAT^{SD,T}_{snt,Px}$ value maintained in the source node's DRS table. Depending on the difference between the two values, N_S updates the path quality values, ($TRV^{T}_{p_x}$ and $ABW^{T}_{P_x}$). It updates the value of $TRV^{Tcur}_{p_x}$ based on Equation 7.1.

$$TRV^{Tcur}_{p_x} = \begin{cases} TRV^{Tcur-1}_{px} - \triangle DFB_{fal} & ,DAT^{SD,T}_{snt,Px} - DAT^{SD,T}_{suc,Px} > THR_{DFB} \\ TRV^{Tcur-1}_{px} + \triangle DFB_{suc} & ,DAT^{SD,T}_{snt,Px} - DAT^{SD,T}_{suc,Px} < THR_{DFB} \end{cases} \tag{7.1}$$

where $\triangle DFB_{fal}$ is an increment value that represents the percentage of failed data packets, $\triangle DFB_{suc}$ is an increment value that represents the percentage of successfully received data packets. $\triangle DFB_{fal}$ and $\triangle DFB_{suc}$ are calculated using Equation 7.2 and Equation 7.3, respectively. THR_{DFB} is a DFB packet threshold value that is determined experimentally in Section 7.6.

$$\triangle DFB_{fal} = \left| \frac{DAT^{SD,T}_{snt,Px} - DAT^{SD,T}_{suc,Px}}{DAT^{SD,T}_{snt,Px}} \right| \tag{7.2}$$

$$\triangle DFB_{suc} = \left| \frac{DAT^{SD,T}_{suc,Px}}{DAT^{SD,T}_{snt,Px}} \right| \tag{7.3}$$

N_S updates the value of ABW^{Tcur}_{Px} using Equation 7.4.

$$ABW^{Tcur}_{Px} = Path\_Expected\_Transmission\_Ratio \times ABW^{Tcur-1}_{Px} \tag{7.4}$$

where, the $Path\_Expected\_Transmission\_Ratio$ is calculated by using Equation 7.5. Equation 7.5 is based on Equation 4.9 in Chapter 4.

$$Path\_Expected\_Transmission\_Ratio = \frac{DAT^{SD,T}_{suc,Px}}{DAT^{SD,T}_{snt,Px}} \tag{7.5}$$

In the worst case, if, by the expiry of $DFBwait$ timeout interval, N_S did not receive any DFB packets from N_D, the $DAT^{SD,T}_{suc,Px}$ value will be zero as there is no value returned from the destination node, thus, $ABW_{P_x}^{T}$ will be zero and the $TRV_{p_x}^{Tcur}$ will be updated based on Equation 7.1, here, the $\triangle DFB$ value will be 1. The PQA algorithm is summarised in Algorithm 7.4.

Algorithm 7.4 The PQA Algorithm

```
function PQA
    Read $DAT_{snt,Px}^{SD,T}$ from the $DRS$ table

    $DAT_{snt,Px}^{SD,T} \leftarrow DAT_{snt,Px}^{SD,T} + 1$

    if $DAT_{suc,Px}^{SD,T}$ ==1 then
        Initiates a $DFB$ timer ($DFBwait$)
    end if
    if $DFBwait$ finishes then
        if $N_S$ did not receive $DFB$ packet then

            $DAT_{suc,Px}^{SD,T} \leftarrow 0$
        else
            Read $DAT_{suc,Px}^{SD,T}$ from the $DFB$ packet
        end if
        Read $TRV_{p_x}^{Tcur-1}$ and $ABW_{P_x}^{T}$ from the $NNI$ table
        if ($DAT_{snt,Px}^{SD,T}$ -$DAT_{suc,Px}^{SD,T}$)>$THR_{DFB}$ then

            $\triangle DFB_{fal} = \left| \frac{DAT_{snt,Px}^{SD,T} - DAT_{suc,Px}^{SD,T}}{DAT_{snt,Px}^{SD,T}} \right|$

            $TRV_{p_x}^{Tcur-1} - \triangle DFB_{fal} \leftarrow TRV_{p_x}^{Tcur}$
            $Link\_Expected\_Transmission\_Ratio = \frac{DAT_{suc,Px}^{SD,T}}{DAT_{snt,Px}^{SD,T}}$
            $ABW_{P_x}^{T} = Link\_Expected\_Transmission\_Ratio \times Traffic\_Transmission\_Rate$
            $DAT_{snt,Px}^{SD,T} \leftarrow 0$
      return False
        else

            $\triangle DFB_{suc} = \left| \frac{DAT_{suc,Px}^{SD,T}}{DAT_{snt,Px}^{SD,T}} \right|$

            $TRV_{p_x}^{Tcur-1} + \triangle DFB_{suc} \leftarrow TRV_{p_x}^{Tcur}$
            $Link\_Expected\_Transmission\_Ratio = \frac{DAT_{suc,Px}^{SD,T}}{DAT_{snt,Px}^{SD,T}}$
            $ABW_{P_x}^{T} = Link\_Expected\_Transmission\_Ratio \times Traffic\_Transmission\_Rate$
```

Algorithm 7.4 The *PQA* Algorithm (continued)

```
            $DAT^{SD,T}_{snt,Px} \leftarrow 0$
      return False
        end if
    end if
      return True
end function
```

7.4.4 DFB Packet Creation (DPC) Algorithm

The *DPC* algorithm is executed by the destination node, N_D, and it counts data packets received at N_D, records this count into a feedback (DFB) packet, and sends the DFB packet to the source node during the data transmission phase. As shown in Algorithm 7.5, upon the reception of the first data packet via a path, P_x, N_D starts to count the data packets received via this path, it maintains a counter recording this count in the $DAT^{SD,T}_{suc,Px}$ field of the DRS table. For this, N_D counts all the incoming data packet from the source node, N_S, via P_x during a period T. The period T starts from the first received data packet's sending time, T_{trn}, that arrives in the data packet header and ends after a predefined time, $DFBwait$, expires (this value is tested experimentally in Section 7.6).

Algorithm 7.5 The *DPC* Algorithm

```
procedure DPC(data)
    Read $DAT^{SD,T}_{suc,Px}$ from the $DRS$ table
    $DAT^{SD,T}_{suc,Px} \leftarrow DAT^{SD,T}_{suc,Px} + 1$
    if $DAT^{SD,T}_{suc,Px}$ ==1 then
        $T_{str} \leftarrow T_{trn}$
    end if
    Wait for $DFBwait$
    Creates a $DFB$ packet
    Write $DAT^{SD,T}_{suc,Px}$ in the $DFB$ packet
    if There is a different path to the source node then
        Send the $DFB$ to $N_S$ using different path
    else
        Send the $DFB$ to $N_S$ via the same path
    end if
    $DAT^{SD,T}_{suc,Px} \leftarrow 0$
end procedure
```

After the T period, N_D creates a DFB packet and writes the $DAT^{SD,T}_{suc,Px}$ value in a DFB header field. Then, N_D sends the DFB to N_S via a different path if N_D has found another trusted path to N_S in its routing table. Otherwise, N_D sends the packet via the same path. Then, N_D resets the $DAT^{SD,T}_{suc,Px}$ counter to start counting the received packets for a new period.

7.5 SAV Solution Operations

This section describes the operations of the SAV solution. The operations differ in terms of the node types in which they are executed. The operations can be classified into SAV source node operations, SAV intermediate node operations, and SAV destination node operations. In this section, we only focus on the source and destination node operations, as the intermediate node operations are identical to the ones used in $TADLV2$.

7.5.1 The SAV Source Node Operations

The SAV source node operations differ in terms of the phase the node is in (path discovery phase or data transmission phase). The operations can be classified into SAV source node operations during the path discovery phase, hereafter referred to as the SAV source node path discovery components, and SAV source node operations during data transmission phase, hereafter referred to as the SAV source nodes data transmission components.

During the path discovery phase, when a source node, N_S, has traffic to send to a destination node, N_D, the $TADLV2$ protocol is used to find multiple path linking N_S to N_D. Once multiple path are discovered, N_S executes the SAV source nodes path discovery components as follows. As shown in Algorithm 7.6, N_S processes the traffic based on its priority. Low priority traffic is transmitted using the Best Path Selection (BPS) algorithm, which has been explained in Chapter 5. BPS is used to select the most trustworthy path, from the discovered paths. High priority traffic is transmitted using the QPS algorithm (explained in Section 7.4.1). Given the required bandwidth, BW_{req}, which is typically determined based on the application generating the traffic to be forwarded. QPS selects one path or a set of paths that together could satisfy the required bandwidth.

Once a path or a set of paths are selected, N_S allocates and sends the traffic via the selected paths by using the QTA algorithm (Section 7.4.2). When the data packets are sent, the data transmission phase starts. Otherwise, if no path or no set of sufficient paths are available, a new $RREQ$ will be transmitted in a larger search area than the one just used (as explained in $TADLV2$). If the previously used search area is already the largest area, Area-L, then N_D is considered to be an unreachable destination.

Algorithm 7.6 The SAV Source Node Operations During Path Discovery Phase

```
procedure SAV-S-Discovery($BW_{req}, data, DiscoveredPaths, IP, SAS_{exs}, Priority$)
    if $Priority$ == Low then
        BPS($IPaddress, DiscoveredPaths, SAS_{exs}, 1$)
    else
        $SelectedPaths \leftarrow$ QPS($DiscoveredPaths, BW_{req}, IPaddress, SAS_{exs}$)
        if $SelectedPaths$ != Null then
            QTA($SelectedPaths, data$)
            Transmit traffic over $SelectedPaths$
            Exit
        else
            if $SAS_{exs}$ == Area-$L$ then
                $N_D$ is an unreachable destination
                Exit
            else
                $SAS_{exs} \leftarrow SAS_{exs} + 1$
                $DiscoveredPaths \leftarrow$ TADLV2($SAS_{exs}$)
            end if
        end if
    end if
end procedure
```

During the data transmission phase, as shown in Algorithm 7.7, N_S invokes the PQA algorithm, explained in Section 7.4.3. In PQA, N_S counts the number of packets sent via each selected path and assesses the selected path quality based on the data carried in each DFB packet. If the path quality cannot satisfy the traffic requirement, N_S will restart the path discovery process.

Algorithm 7.7 The SAV Source Node Operations During Data Transmission Phase

```
function SAV-S-Transmission($data$)
    $Flag \leftarrow$ PQA
    return $Flag$
end function
```

7.5.2 The SAV Destination Node Operations

During a path discovery phase, the destination node may receive a $RREQ$ packet and when this happens, it invokes the $TADL$ destination node operations, explained in Chapter 5, whereas, during the data transmission phase, the destination node may receive data packets from the source node and when this happens, as shown in Algorithm 7.8, it invokes the DPC algorithm, explained in Section 7.4.4. The DPC algorithm counts all the data packet successfully received from the source node via a path, P_x, during the period T. It then constructs a DFB packet, writes the counted number of packets into the DFB header and sends the DFB packet to the source node.

Algorithm 7.8 The *SAV* Destination Node Operations During Data Transmission Phase

```
procedure SAV-D-TRANSMISSION($data$)
    DPC($data$)
end procedure
```

7.6 Simulation Study

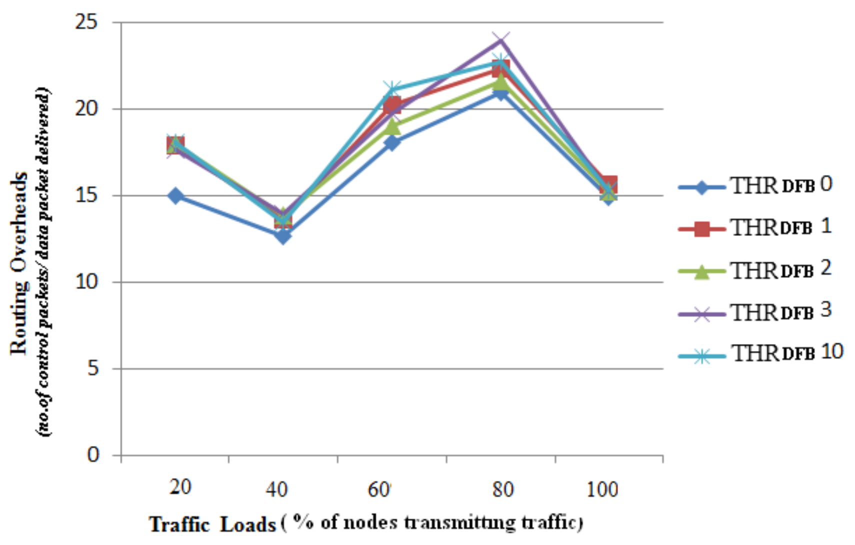
The performance of SAV is evaluated using a simulation study. This section determines the result of this simulation study. It first describes how the threshold values are determined using simulation, and then presents and analyses the results of the SAV solution against those from $TADLV2$.

7.6.1 Determining Thresholds

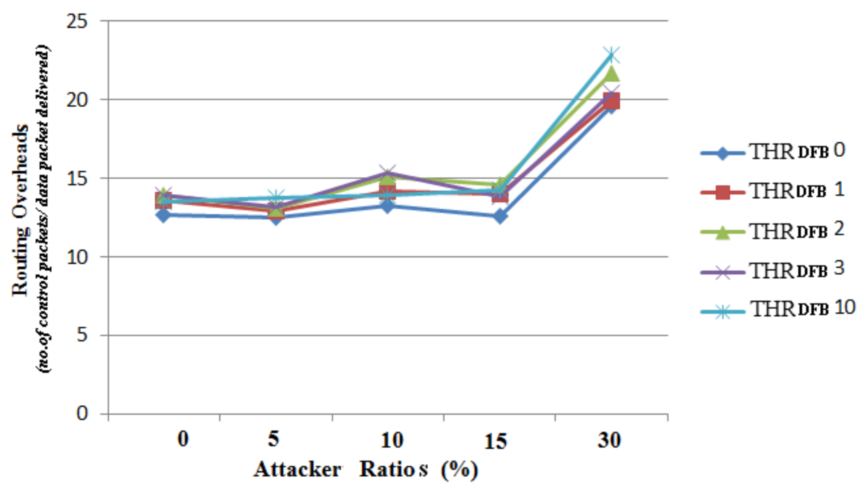
The two threshold values, THR_{DFB}, and $DFBwait$, are used in SAV during the data transmission phase as described in Section 7.4. THR_{DFB} is used by N_S, as described in Section 7.4.3 and $DFBwait$ is used by N_S and N_D, as described in Sections 7.4.3 and 7.4.4. These parameter values are used to govern the adjustments of the path quality values, TRV_{px}^{Tcur} and $ABW_{P_x}^{T}$.

7.6.1.1 Determining THR_{DFB} Value

The simulation results, shown in Figure 7.2, are to determine an appropriate value for THR_{DFB}. THR_{DFB} is used by N_S to determine and to adjust the path quality during a data transmission phase, as described in Section 7.4.



(a) Routing Overheads vs. Network Traffic Loads



(b) Routing Overheads vs. Attacker Ratios

Figure 7.2: Routing Overheads vs. Network Traffic Loads and Attacker Ratios with Different THR_{DFB} Values

Five values of THR_{DFB} have been investigated. These are 0, 1, 2, 3 and 10 packets. The results are marked as $THR_{DFB}0$, $THR_{DFB}1$, $THR_{DFB}2$, $THR_{DFB}3$, and $THR_{DFB}10$ we examine how the different THR_{DFB} values affect the routing overheads. Two simulations are carried out. As shown in Figure 7.2, in the first figure, the results are studied when varying the traffic load ratios in the network. The traffic loads used are respectively 20%, 40%, 60%, 80%, and 100% of the nodes transmit traffic. In this simulation the attacker ratio used is 0% and the pause time is 300 s. In the second figure, the results are studied when varying the attacker ratios in the network. The attacker ratios of 0%, 5%, 10%, 15%, and 30% are used. In these results, the traffic load used is that 40% of the nodes are source node and the pause time used is 300 s. All other simulation parameter values that are used in the simulation studies are as specified in Section 4.6.

As shown from both figures, $THR_{DFB}0$ offers the lower routing overhead in the network. Therefore, $THR_{DFB} = 0$ packet is chosen. This means that no difference between the number of received data packet at the destination node and the number of data packets sent by N_S is accepted.

7.6.1.2 Determining $DFBwait$ Value

The simulation results, shown in Figure 7.3, are to determine an appropriate value for $DFBwait$. $DFBwait$ is used to determine the period of time T, that a destination, N_D, and a source node, N_S should wait to assess the quality of the path linking the two nodes. In other words, N_D counts the number of data packets received during $DFBwait$ seconds, after the arrival of the first data packet. It then creates a DFB packet and sends it to N_S. On the source node side, N_S counts all the data packets sent to N_D via a path, P_x, during $DFBwait$ seconds, and waits for $DFBwait$ time for a DFB packet to arrive from N_D, which contain the number of data packets successfully received during this period.

Four values of $DFBwait$ are investigated. They are 1, 4, 8, and 12 s. The $DFBwait$ values marked DW_1, DW_4, DW_8, and DW_{12} are plotted in Figure 7.3. These results show the effect of the different $DFBwait$ values on the routing overheads. The results are collected with varying the attacker ratios in the network. The attacker ratios used are 0%, 5%, 10%, 15% and 30%, respectively. In these results, the traffic load used is that 40% of the nodes are source node and the pause time used is 300 s. All other simulation parameter values that are used in the simulation studies are as specified in Section 4.6. From the results, it can be seen that DW_8 offers the lowest routing overheads, therefore, the $DFBwait$ value is set to 8 s.

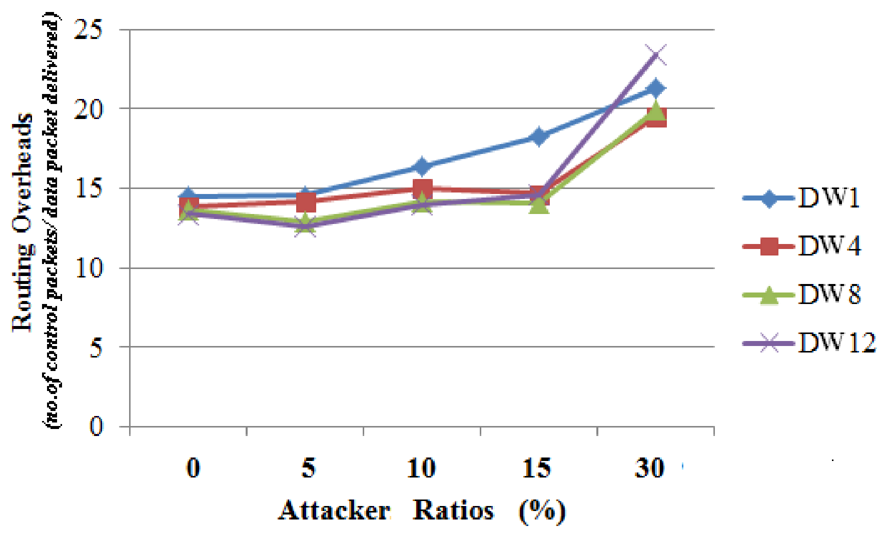


Figure 7.3: Routing Overheads vs. Attacker Ratios with Different $DFBwait$ Values

7.6.2 Simulation Results and Discussions

This section presents and discusses the simulation results of the SAV solution. We will compare the results of the SAV solution against that of $TADLV2$ (explained in Chapter 6) in terms of routing overheads, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$). SAV uses $TADLV2$ to search for multiple path between the communication nodes. However, during the path selection and traffic allocation stages, SAV is different from $TADLV2$ in four aspects. Firstly, SAV uses a single path or multiple path to deliver high priority traffic depending on the traffic bandwidth requirement versus path quality. The selection of the paths is based on the path qualities (path available bandwidths and trust values) which are estimated during the path discovery process, whereas, $TADLV2$ uses a single and most trustworthy path to deliver traffic between a pair of communication nodes. Secondly, when the multiple path delivery service is used, SAV allocates the traffic to the selected multiple path based on their path qualities, whereas $TADLV2$ does not have this functionality. Rather, $TADLV2$ allocates all the traffic to the single path selected. Thirdly, for paths that have low trust values, i.e. paths that are classified as untrustworthy, SAV duplicates packets on these untrustworthy paths to increase delivery success, whereas, $TADLV2$ only allocates one copy of each packet on the selected path even if the path trust value is low. Finally, during the data transmission phase, SAV uses the feedback from the destination node to update the path quality values and to adjust the traffic allocation and, if necessary, to reselect a path or a set of paths that will be used to deliver high priority traffic, whereas, $TADLV2$ does not use destination node feedback.

The purpose of this investigation is to study if the additional four functionalities of SAV brings any benefits in terms of reducing routing overhead and increasing PDR. We study the effects of varying traffic loads and attacker ratio levels to the SAV performance, in terms of the routing overhead, packet delivery ratios ($PDRs$), and average end-to-end packet delivery delays ($DLYs$). No results for varying mobility levels are included for SAV because varying mobility levels had the same effect on SAV and $TADLV2$. Two sets of simulation results are presented and analysed. Set 1 (Section 7.6.2.1) results, as shown in Figure 7.4, shows the effects of varying network traffic loads on the SAV performance. The traffic loads are measured in terms of the percentage of nodes in the network that are source nodes, i.e. traffic is generated by 20%, 40%, 60%, 80%, and 100% of the nodes, respectively. 20% of source nodes are used to simulate a lightly loaded network. 40% and 60% of source nodes are used to simulate a medium loaded network. 80% and 100% of the source nodes are used to simulate a highly loaded network.

Set 2 (Section 7.6.2.2) results, as shown in Figure 7.5, show the effects of packet dropping attacks on the SAV performance. The investigation is carried out by setting 0%, 5%, 10%, 15%, and 30% of the nodes in the network as attackers. All other simulation parameter values used in the two sets of simulation runs are set as specified in Section 4.6.

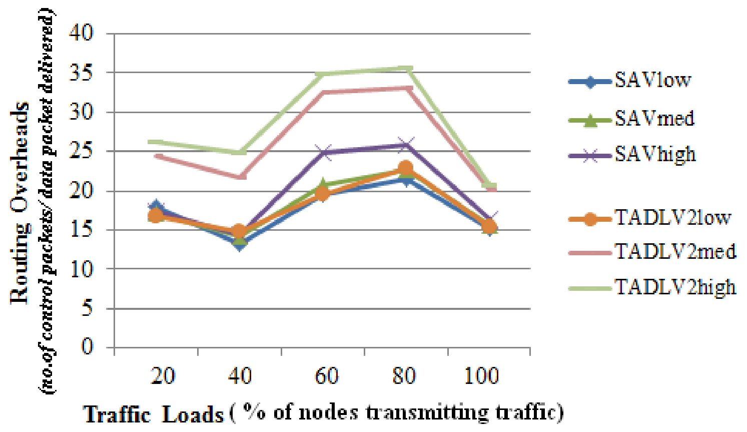
The $TADLV2$ is examined with varying required number of paths, NPS_{req}, between the communication nodes. SAV is examined by varying required bandwidth, BW_{req}. The $TADLV2$ traffic is marked as $TADLV2low$, $TADLV2med$, and $TADLV2high$ in the figures. The SAV traffic is marked as $SAVlow$, $SAVmed$, and $SAVhigh$ in the figures, where *low* represents a low BW_{req} value, *med* represents a medium BW_{req} value, and *high* represents a high BW_{req} value. These values are set as 1, 3, and 4 packets/s, respectively. Each result is obtained by averaging 30 simulation runs and the confidence level is $<= 2.86$.

7.6.2.1 SAV vs. TADLV2: Varying Traffic Loads

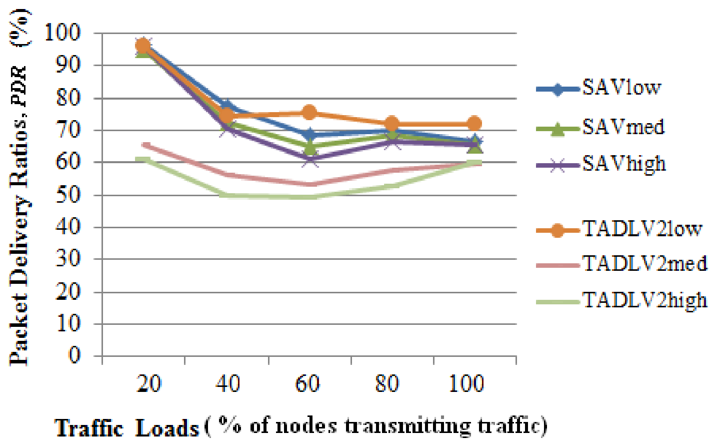
The first set of results is plotted in Figure 7.4, where we examine the effects of changing network traffic loads on the routing overheads, $PDRs$, and $DLYs$. In this investigation, the attacker ratio is set to 0% and the pause time is set to 300 s.

Figure 7.4(a) shows the routing overhead of SAV and $TADLV2$ versus traffic loads. From the figure, we can make three observations. The first observation is that when the network load increases, the routing overheads increase with both SAV and $TADLV2$ (except for 100% traffic load). The reason for this result has been explained in Section 5.5.2.2. However, in the heavy loaded network, when 100% of the nodes are transmitting traffic, the routing overheads decrease. This is because, in the heavily loaded network, the network gets congested and the packets (control and data) are dropped causing fewer packets being transmitted in the network, lost packets leads to lowers level of routing overhead. The second observation is that for the whole range of traffic loads investigated, SAV outperforms $TADLV2$ except for $TADLV2low$, where $TADLV2low$ results are similar to SAV results. This is because, in the case of the SAV solution, the distribution of the traffic across multiple path can decrease the chance of the network being congested. $TADLV2$ discovers multiple path but only uses one of the discovered paths for each traffic delivery. In the case of $TADLV2low$, only one path is required. The chance of finding a single path to successfully transmit the traffic is higher than the other $TADLV2$ load case. So, in this case, the routing overhead is similar to that of SAV. The third observation is that for both solutions, as the traffic requirements increase the routing overhead increases. The increase in the routing overhead indicates that more control packets are used to discover the required paths or to satisfy the required bandwidth.

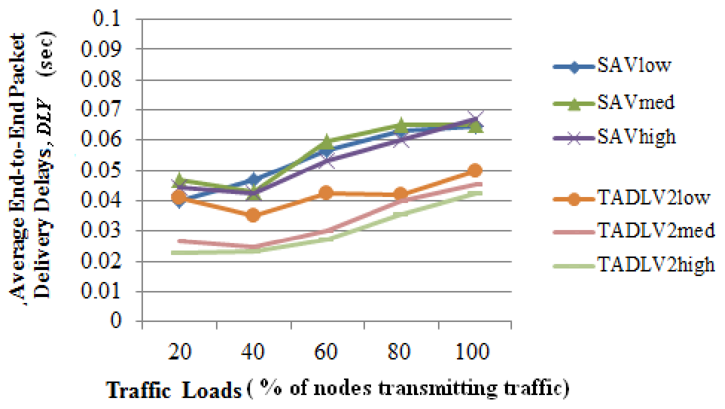
Figure 7.4(b) shows the $PDRs$ versus traffic loads for the two solutions. From the figure, it can be seen that, firstly, in all the cases, when the traffic load increases, the PDR values decrease, but the PDR values level off when the traffic load reaches to a saturation point when the network is getting congested. This saturation point is dependent on the solution used and the bandwidth requirements imposed by the user traffic. For example, for all the cases of SAV and $TADLV2low$, the saturation point is when 60% of the network nodes are transmitting. However, this saturation point is reduced to 40% for the case of



(a) Routing Overheads vs. Network Traffic Loads



(b) Packet Delivery Ratios vs. Network Traffic Loads



(c) Average End-to-End Packet Delivery Delays vs. Network Traffic Loads

Figure 7.4: Routing Overheads, Packet Delivery Ratios, and Average End-to-End Packet Delivery Delays vs. Network Traffic Loads for *SAV* and *TADLV2*

$TADLV2$ when the bandwidth requirements are at the medium and high levels. These results are consistent with the results shown in Figure 7.4(a). The higher the traffic load and/or the higher the bandwidth requirement, the higher the routing overhead as more paths will be required. A higher level of routing overhead will make the network get into a congested state. Secondly, when the requirement is at the lowest level, $TADLV2$ gives similar PDR performance to the one from SAV in the low to high cases investigated. When the requirement imposed by the traffic is low, it is easier to discover a path that could satisfy the requirement, and in addition, as it does not incur the overhead for multiple path delivery, so $TADLV2$ offers a similar PDR to its SAV counterpart. However, when the requirement is at the medium and high level, $TADLV2$ produces much worse PDR performance than all the other cases. This can be attributed to two reasons. The first is that SAV can harvest the bandwidth offered by multiple path to deliver a high priority traffic flow and distribute the traffic over the multiple path. On the other hand, $TADLV2$ only uses a single path to deliver the traffic. The second reason is that, in the case of SAV, using the feedback, can adjust the injection of traffic based on the network condition and this can decrease the number of packets lost, increasing the PDR. For this reason, SAV performs much better, in terms of both routing overhead and $PDRs$, when the traffic requirement goes beyond medium level.

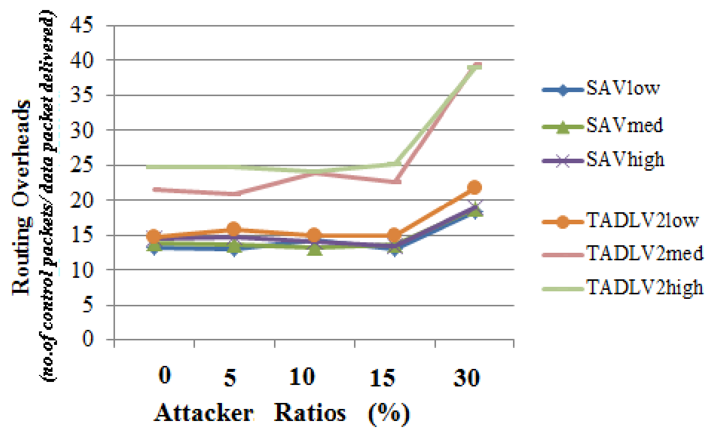
Figures 7.4(c) shows the $DLYs$ versus traffic loads. From the figure, it can be seen that, as traffic load increases, the $DLYs$ also increase, and this is the case for both solutions under all the requirement settings. This result is intuitive, as when traffic load increases, the network will get busier, and the delays suffered by the packets will increase accordingly. Also, from the figure, it can be seen that, in all cases of SAV, the delays are higher than the delays for the case of $TADLV2$. If we examine these results closely in conjunction with the results shown in Figure 7.4(b), we can make the following two major observations. Firstly, when the bandwidth requirement is low, $TADLV2$ performs better than SAV in terms of $DLYs$ and performs the same in terms of $PDRs$. However, when the requirement is high, the DLY value for $TADLV2$ goes lower and so is its PDR value. This is because, when the requirement goes higher, more packets will be dropped (as reflected by the lower PDR values) and the dropped packets are not counted for when the DLY value is calculated. Generally, as shown in these two figures, the changes in $PDRs$ and in $DLYs$ in the case of SAV are not as sensitive to the increase of bandwidth requirement as in the case of $TADLV2$. This means that SAV can better support different levels of bandwidth requirements, and when the bandwidth requirement is high, SAV offers a higher level of delivery reliability than $TADLV2$.

Figure 7.4 observations indicate that distributing the traffic over multiple path, adjusting the injection of control packet, and adjusting the path quality during the data transmission phase in SAV solution can significantly reduce the routing overhead injected into the underlying network even under a highly loaded network. The reduction in the routing overhead improves the performance of the SAV solution in terms of $PDRs$, but this is at the cost of a higher DLY.

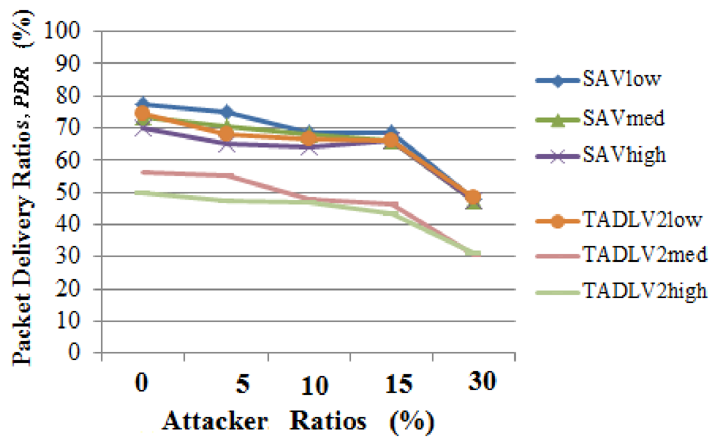
7.6.2.2 SAV vs. TADLV2: Varying Attacker Ratios

The second set of figures, shown in Figure 7.5, show the effects of attacker ratios on the routing overheads, $PDRs$, and $DLYs$. In this set of figures the traffic load used is 40% and the pause time is 300 s.

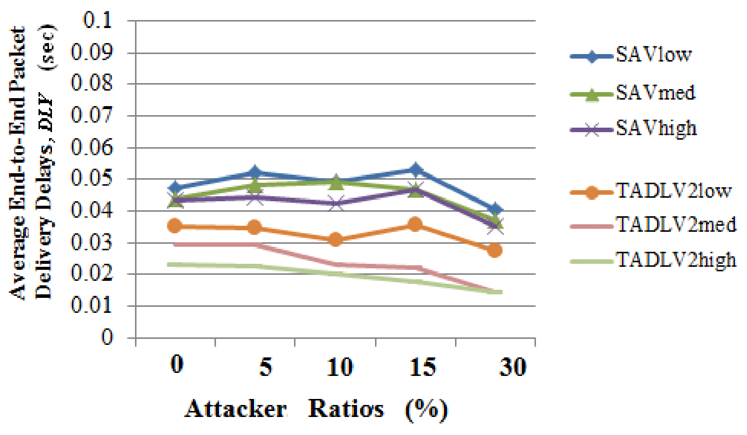
Figures 7.5(a) and 7.5(b), respectively, show the routing overhead and $PDRs$ versus attacker ratios. When the attacker ratio increases, the routing overhead are largely unchanged before the attacker ratio reaches to 15% when they increase markedly. The figure also shows that the increase is sharper for the case of $TADLV2med$ and $TADLV2high$. This means that when the attacker ratio reaches to a certain value, packet loss will trigger more control packets to be dispatched to discover more paths, resulting in a higher level of routing overhead. As $TADLV2$, unlike SAV, cannot make use of untrustworthy paths, more route discovery instances are triggered, resulting in a worse performance in the case of $TADLV2med$ and $TADLV2high$. Also, as expected, the PDR values for all the SAV and $TADLV2$ cases, the PDR decreases are fairly moderate before the attacker ratio reaches 15%. This is because when the routing overheads increase beyond a certain point, the network is congested and this will speed up the decrease of PDR. In other words, under a given parameter value settings, when the attacker ratio reaches beyond 15%, the network is in a congested state. When the attacker level is higher more packets will be dropped. This should help to reduce the chance of the network being congested. However, on closer examination, we can see that a higher attacker ratio will lead to more paths being rated as untrusted (in the case of SAV), or fewer trusted paths. This means that more packet duplications will be used, and more paths will need to be discovered,



(a) Routing Overheads vs. Attacker Ratios



(b) Packet Delivery Ratios vs. Attacker Ratios



(c) Average End-to-End Packet Delivery Delays vs. Attacker Ratios

Figure 7.5: Routing Overheads, Packet Delivery Ratios, and Average End-to-End Packet Delivery Delays vs. Attacker Ratios for *SAV* and *TADLV2*

leading to more path discovery processes.

With regard to $DLYs$, as shown in Figure 7.5(c), as the attacker ratio increases, the delays decrease. This is due to the fact that a higher attacker ratio will lead to more packets being dropped, and the dropped packets are not considered in the packet delay calculation. Consistent with the results presented in Figure 7.4, SAV introduces more packet delivery delays in comparison with $TADLV2$, as the latter delivers fewer packets through the network successfully.

The above results show that, under various traffic load and attacker ratio conditions, SAV outperforms $TADLV2$, in terms of traffic delivery reliability, especially when the bandwidth requirements are on the high end. This indicates that the approaches of multi-path delivery, packet duplication over untrustworthy paths, and adjusting the path quality during the data transmission phase, which have been implemented in SAV, are more effective than the single path approach taken in $TADLV2$, in supporting high priority traffic deliveries.

7.7 Chapter Summary

This chapter has presented the design and simulation of a novel solution, SAV to improve the delivery of high priority traffic to their destinations in the presence of node mobility and packet dropping attacks. The solution uses two different delivery services to deliver low and high priority traffic, respectively, and for the high priority traffic delivery, it allocates the traffic based on the available bandwidth of each path and its trust value estimated during the paths discovery phase. By using the destination node feedback mechanism, a source node can verify and adjust the quality value assigned to each path during the data transmission phase. This not only allows the source node to dynamically adjust the value of traffic allocated to each path but also can reduce the level of trust the source node places on the intermediate nodes.

We have used simulations to evaluate the effectiveness of these ideas built in SAV and compared the results from SAV to $TADLV2$. $TADLV2$ uses the most trustworthy single path to deliver traffic. Our simulation results show that SAV is more effective in reducing the number of control packets injected into the underlying network, and in increasing packet delivery reliability. It outperforms the $TADLV2$ protocol, the most relevant related work, in medium and high bandwidth requirement conditions.

Chapter 8

Conclusions and Future Works

The focus of this thesis has been to investigate how to best support the delivery of high priority traffic in the presence of node mobility and packet dropping attacks in MANETs. This chapter summarises the work presented in this thesis, highlighting the contributions and discoveries from this research, and gives recommendations for future work.

8.1 Thesis Conclusions

Background Research

In this thesis, the MANET characteristics were reviewed, the application scenarios were discussed to determine the need for *QoS* in MANETs, and the security threats were investigated to understand their impacts on *QoS*. Then, the related work was studied and critically analysed to identify their respective strengths and limitations, so that in our design we could maintain the strengths while overcoming the weaknesses. Then the challenging issues in achieving *QoS* in MANETs were discussed, the requirements and the measures to satisfy them are specified and identified. Based on the requirements and measures, we presented a framework called an Adaptive Trust-Aware Location-based (*ATL-QoS*) framework.

The *ATL-QoS* framework is designed to optimise high priority traffic delivery in MANETs containing packet dropping attackers. It has the following four features: (a) it can discover multiple path and identify more trustworthy paths during the path discovery process and do so with low bandwidth cost imposed in the network and low computational cost imposed on all the nodes in the network; (b) it can use single or multiple (trusted or untrusted) paths to deliver high priority traffic and when using untrusted paths, it uses packet duplication to increase high priority traffic delivery success; (c) it can dynamically adjust the injection of high priority traffic in adaptation to the fluctuated available bandwidth and threat level of the used paths; and (d) it uses an end-to-end feedback mechanism to allow a source node to validate the path quality assessed during the path discovery phase and this allows the source node to re-adjust the amount of traffic assigned to the paths concerned and allow the source node to detect cheating of path available bandwidth by intermediate nodes, thus reducing trust placed on the intermediate nodes.

The *ATL-QoS* framework is comprised of two planes, a control plane and a data plane. The control plane is responsible for (1) multiple path discovery, (2) the selection of a set of more trustworthy paths from the discovered paths, (3) the allocation of traffic to the set of selected paths for transmissions and (4) the feedback mechanisms and the verification of path quality during the data transmission phase. Function (1) is implemented in the $TADL$ (early version) and $TADLV2$ (improved version) protocols. Functions (2), (3), and (4) are implemented in the SAV solution. The data plane is responsible for packet transmission, forwarding and reception.

TADL: A Trust-Aware Dynamic Location-Based Multiple Path Discovery Protocol

The $TADL$ protocol is designed to discover multiple trustworthy paths for routing traffic towards a destination node while minimising overhead costs. The following three measures are taken in the design of $TADL$. Firstly, it uses a destination location-based path discovery approach to discover paths. With this approach, route request packets are only broadcast within a defined search area in the network reducing the volume of traffic the network has to handle. Secondly, it dynamically adjusts the size of the starting search area in which route discovery packets are broadcast in adaptation to the underlying network conditions, thus effectively reducing the routing overhead imposed on the network. Thirdly, it assesses and uses nodes' trust values to govern the selection of nodes during path formation, and to reduce any overhead introduced, a direct trust model is used.

To demonstrate the effectiveness and efficiency of the $TADL$ protocol, it is implemented using the *NS-2* simulation package, evaluated and compared against the LAR protocol, the most relevant protocol in the literature. The simulation study shows that $TADL$ provides a better performance than LAR in terms of reducing routing overheads, decreasing the average end-to-end packet delivery delays, and increasing packet delivery ratios (except when the network nodes' mobility level is high). The study also shows that $TADL$ performs best in a stationary network or a network with a low to medium level of node mobility. However, when the network nodes' mobility level is high, the number of control packets, produced by $TADL$, increases sharply. Based on the findings of the simulation study of $TADL$, it is found that the first-discover-then-select approach to path discovery used in $TADL$ is not effective in a highly mobile network. This finding along with other ideas generated during the simulation study of $TADL$ has motivated us to re-designed the $TADL$ protocol, producing the $TADLV2$ protocol.

TADLV2: An Improved Version of TADL

$TADLV2$ is designed to overcome the weaknesses identified in the $TADL$ protocol. $TADLV2$ is different from $TADL$ in two aspects. Firstly, it uses different algorithms to discover node-disjoint paths based on the neighbouring nodes' mobility levels. When the mobility level is low or medium, the source node control method, as the one used in $TADL$, is used, i.e. the source node is responsible for node-disjoint path discovery. However, when the mobility level is high, an intermediate node control method is used, in which intermediate nodes are involved in the node-disjoint path discovery. Secondly, it uses a mobility-based adaptive starting search area adjustment algorithm to dynamically adjust the starting search area size based on the underlying network conditions. This algorithm, along with the new approach to path discovery, enables multiple node-disjoint paths to be discovered with less routing overhead in all mobility level situations.

The performance of the $TADLV2$ protocol has been evaluated and compared against that of $TADL$. The simulation results show that the measures taken in $TADLV2$ are effective in reducing overheads in multiple path discoveries. However, there are still three aspects we can improve on the $TADLV2$ design. These are: (a) $TADLV2$ assumes more trust on the intermediate nodes, and as an intermediate node could be a packet dropping attacker, and it may lie about its available bandwidth value, so measures should be taken to counter this threat; (b) $TADLV2$ searches for multiple path, but it only uses one path to deliver the traffic, and as it is not always possible to find a single path that is both trustworthy and has sufficient bandwidth to deliver traffic, improvement should be made to allow multiple path, including those that are not trustworthy, to deliver traffic; (c) $TADLV2$ treats all the traffic in the same way, so measures should be taken to differentiate the priority of traffic and to deliver high priority traffic in a more reliable manner. The SAV solution, described below, has been added into the *ATL-QoS* framework to realise these improvements.

SAV: Path Selection, Traffic Allocation, and Path Verification to Support High Priority Traffic Delivery

To support the QoS requirements of high priority traffic effectively, it is necessary to satisfy the minimum user/application level bandwidth requirements. In the $TADLV2$ protocol, after discovering paths, trust values are used to govern the selection of the most trustworthy path from the discovered paths to route traffic. However, due to the fluctuating available bandwidth, it is not always possible to find a single path that is both trustworthy and has the required bandwidth to support the delivery of high priority traffic. In addition, the estimated available bandwidth and trust value of a discovered path may not be accurate due to node mobility and it has used the input from the intermediate nodes along the path, and the intermediate nodes may lie about their available bandwidth values.

To address these weaknesses, a path Selection, traffic Allocation, and path Verification (SAV) solution was designed. The package consists of four functional blocks which are integrated with $TADLV2$ in the *ATL-QoS* framework. These functional blocks, in conjunction with $TADLV2$, make the *ATL-QoS* framework have the following capabilities. It provides differential service to the delivery of high and low priority traffic. For high priority traffic, depending on the bandwidth requirement of the traffic to be delivered, the available bandwidth provided by the paths discovered, and the underlying network conditions, including node mobility levels, the source node can harvest the bandwidth provided by multiple path, trusted or untrusted, and spread the traffic over the paths. Packet duplication is used for packet delivery over untrusted paths. In this way, both load balancing and a better use of the discovered bandwidth for a more reliable packet delivery can be achieved. In addition, it uses a destination node feedback mechanism to detect any possible cheating of bandwidth capacity along a path by an intermediate node and to allow a source node to adjust the quality of the path (bandwidth and trust values) during the data transmission phase.

A simulation study has been carried out to evaluate the effectiveness of the ideas used in SAV and the results are compared with that from $TADLV2$, i.e. before the SAV functions are added. The results show that SAV outperforms $TADLV2$ in terms of reducing the number of control packets injected into the underlying network, and increasing the $PDRs$.

ATL-QoS Generalisation

The ATL-QoS framework is applicable to any networks with mobility support or a dynamically changing topology such as mobile wireless sensor network ($MWSN$), Vehicular Ad-hoc NETworks ($VANETs$) and internet-based Mobile Ad hoc NETworks ($iMANETs$). Those networks are all battery powered, so saving energy is hugely beneficial. They all use different types of data, with different priorities and different QoS requirements, transported in the network. In addition, the network is vulnerable to a range of security attacks that may affect the QoS. So they all need to consider these attacks when designing QoS solution for MANETs.

8.2 Suggestions for Future Research

The following presents five recommendations for future research.

Differentiating Malicious Attackers from Overloaded Nodes

The loss of data packets may be caused by two factors: (1) malicious intermediate nodes, i.e. packet dropping attackers; (2) non-malicious factors, such as node mobility that causes link breaks, and intermediate node buffer overflow. Packet loss caused by different factors may have certain patterns, e.g. the loss caused by malicious nodes is usually to forward control packets as they want to be included in the path. However, packet loss caused by non-malicious events would happen to both data and control packets. We classify a node with buffer overflow as an overloaded node, and an overloaded node drops both data and control packets. $ATL\text{-}QoS$ does not differentiate between these two classes of factors. Differentiating these different factors may allow us to tackle each using different measures, further improve the efficiency of the solution. For example, if the packet loss is caused by non-malicious factors, we should not use packet duplication.

Determining a better data duplication strategy

In *ATL-QoS*, duplicated data packets are sent via a low trusted path, we believe this may help to increase packet delivery reliability. There may be a more efficient way to duplicate the data. For example, data may be split into a number of smaller packets, each containing some extra bits. The extra bits are calculated in such a way that the original packet can be reconstructed given a subset of these smaller packets. The split packets are then transmitted across multiple path. In this way, we may reduce the bandwidth cost by data duplication, but at an extra processing cost by the source and destination nodes.

Estimating the neighbouring node trust

In the research presented in this thesis, it is assumed that the neighbouring node attacker ratio is known by all the nodes. There should be an effective way to estimate the neighbouring node attacker ratios.

Investigating the energy costs of the ATL-QoS Framework

The research presented in this thesis has not considered the energy costs of the proposed solutions. The energy consumed may lead to some nodes' batteries becoming exhausted. This will mean that these nodes will no longer be able to participate in the network. Consequently, some links may break, the topology will change, and the available bandwidth will decrease. All of these may negatively affect *QoS*.

Two different methods can be investigated in this regard. Firstly, node energy can be used for path selection, during the path discovery phase. Intermediate nodes can be selected to form the path based on their current energy levels in addition to their trust values. In this method, intermediate nodes can, with a higher probability, survive for the duration of a data transmission phase. Thus, leading to a better *QoS* in terms of packet delivery ratios. Another method is the route-break prediction method that can be applied during the data transmission phase to predict a link break before it actually happens. Neighbouring nodes' energy levels can be used to predict link failures. Based on this prediction, the source node can discover a new path before the old path breaks. In this way, we may reduce the level of packet loss.

Applying ATL-QoS to an IMANET

The *ATL-QoS* framework has not been evaluated in the context of an Internet-based MANET (IMANET), where a MANET is integrated with the Internet to serve as an access network. This interconnection is usually achieved by using gateways between a MANET and the Internet. *ATL-QoS* could be investigated to determine whether it can perform effectively in this environment. There are several possible areas of investigation. For example, is the *ATL-QoS* framework applicable to the IMANET environment? How can gateway nodes be trusted to act as a bridge between the MANET and the Internet? How can nodes in a MANET be made addressable so that they can be contacted by nodes on the Internet?

Using 3-Dimensional Location-based Approach

The *ATL-QoS* framework composed of wireless nodes randomly distributed in a 2-Dimensional geographical region. We have not considered using a 3-Dimensional geographical region, such as using the *ATL-QoS* framework in building or mountainous area. The location information in a real environment can be acquired by *GPS* demonstrated by a 3-Dimensional coordinate (x, y, z). In a 3-Dimensional real scenario, there will be possible consequences of the shape of the predicted location of N_D and the search area. The predicted location of N_D can be represented by a sphere or ellipsoid and the search area could be a cuboid, cylinder or a pyramid.

Determining Another Path Available Bandwidth and Path Trust Value Estimation Approach

The path available bandwidth and path trust value are estimated based on the weakest link principle. This means that the path trust is the value for the least trustworthy path in the links that forming the path and the path available bandwidth is the value for the least available bandwidth path in the links that forming the path. We want to investigate whether this was the most appropriate approach. For example, for reliability, it can be argued that the product of the probabilities (proportions) of successfully delivered packets would be a more accurate metric.

Investigating Other Attack Behaviours

We assume that when a selfish node receives data or control packets, it simply drops them. In a more nuanced approach, selfish or malicious nodes may have a attempt to drop only as many packets as they can get away with, without being detected or ostracised. More investigation may be done on what impact this might have, how they might be detected, and the impact of other attack behaviours.

Comparing ATL-QoS to LAR

We first analysed the study results of the $TADL$ protocol and compare the results against those from LAR. Then, we compared the performance of the $TADLV2$ protocol against that of the $TADL$ protocol. Finally, we compared the results of the $ATL-QoS$ framework against that of $TADLV2$. A Final comparison of the $ATL-QoS$ framework results against that of LAR is needed.

Investigating Error Rate

In the simulation configuration, the packet/bit error rate is set to 0, as the focus of this research is packet loss due to attackers, node mobility, and congestion at the network layer. It would be interesting to investigate the effects of the error rates on the proposed framework, as caused by noise or interference levels at the physical layer.

In conclusion, the aim of this research to achieve QoS provisioning in a MANET containing packet forwarding attackers has been achieved but there remain other possible avenues of future work.

Bibliography

[1] A. Abdrabou and W. Zhuang. A position-based QoS routing scheme for uwb ad-hoc networks. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 8, pages 3578–3584. IEEE, 2006.

[2] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad hoc networks*, 2(1):1–22, 2004.

[3] Daniel Aguayo, John Bicket, Sanjit Biswas, Douglas SJ De Couto, and Robert Morris. Mit roofnet implementation. 2003.

[4] M. Ali, B. G. Stewart, A. Shahrabi, and A. Vallavaraj. Congestion adaptive multipath routing for load balancing in mobile ad hoc networks. In *Innovations in Information Technology (IIT), 2012 International Conference on*, pages 305–309, March 2012.

[5] H. Y. An, X. C. Lu, Z. Gong, and W. Peng. A cluster-based QoS multipath routing protocol for large-scale MANET. *High Performance Computing and Communications*, pages 321–330, 2005.

[6] V Anitha and Dr J Akilandeswari. Secured message transmission in mobile ad hoc networks through identification and removal of byzantine failures. *arXiv preprint arXiv: 1104.4690*, 2011.

[7] H. S. Arora. *Towards achieving QoS guarantees in mobile ad hoc networks.* PhD thesis, Drexel University, 2003.

[8] Baruch Awerbuch, Reza Curtmola, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. ODSBR: An on-demand secure byzantine resilient routing protocol for wireless ad hoc networks. *ACM Trans. Inf. Syst. Secur.*, 10(4):6: 1–6: 35, January 2008.

[9] H. Badis, K. Al Agha, et al. Quality of service for ad hoc optimized link state routing protocol (QOLSR). 2007.

[10] Helen Bakhsh, Ning Zhang, and Andy Carpenter. TADL: A trust-aware dynamic location-based protocol suite for discovering multiple path in MANETs. In *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, ICDCN '15, pages 2: 1–2: 10, New York, NY, USA, 2015. ACM.

[11] Helen Bakhsh, Ning Zhang, and Andy Carpenter. TADL-V2: An improved trust-aware dynamic location-based adaptation protocol for discovering multiple path in MANETs. *EAI Endorsed Transactions on Ambient Systems*, 15(6), 8 2015.

[12] K. Balakrishnan, Jing Deng, and P. K. Varshney. Twoack: preventing selfishness in mobile ad hoc networks. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 4, pages 2137–2142 Vol. 4, March 2005.

[13] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '98, pages 76–84, New York, NY, USA, 1998. ACM.

[14] J. Biswas, M. Barai, and S. K. Nandy. Efficient hybrid multicast routing protocol for ad-hoc wireless networks. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 180 – 187, nov. 2004.

[15] L. Blazevic, J. Y. Le Boudec, and S. Giordano. A location-based routing method for mobile ad hoc networks. *Mobile Computing, IEEE Transactions on*, 4(2):97–110, March 2005.

[16] G. Bos. QoS support using diffserv. *6th TSConIT*, 2007.

[17] J.N. Boshoff and A.S.J. Helberg. Improving QoS for real-time multimedia traffic in ad-hoc networks with delay aware multi-path routing. In *Wireless Telecommunications Symposium, 2008. WTS 2008*, pages 1–8, April 2008.

[18] Jean-Yves Le Boudec and Milan Vojnovic. Perfect simulation and stationarity of a class of mobility models. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2743–2754. IEEE, 2005.

[19] Josh Broch, David A Maltz, David B Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97. ACM, 1998.

[20] I. D. Chakeres and E. M. Belding-Royer. AODV routing protocol implementation design. In *Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on*, pages 698–703. IEEE, 2004.

[21] L. Chen. *Protocols for supporting quality of service in mobile ad hoc networks*. PhD thesis, University of Rochester, 2006.

[22] DouglasS. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. a high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[23] S. K. Das, A. Mukherjee, S. Bandyopadhyay, K. Paul, and D. Saha. Improving quality-of-service in ad hoc wireless networks with adaptive multi-path routing. In *Global Telecommunications Conference, 2000. GLOBECOM'00. IEEE*, volume 1, pages 261–265. IEEE, 2000.

[24] S.R. Deshmukh and V.T. Raisinghani. Ealbm: Energy aware load balancing multipath routing protocol for MANETs. In *Wireless and Optical Communications Networks (WOCN), 2014 Eleventh International Conference on*, pages 1–7, Sept 2014.

[25] S. Ding. Protocol interoperability in mobile ip enabled mobile ad hoc networks. *Computer Standards & Interfaces*, 31(4):830–845, 2009.

[26] Shunli Ding and Liping Liu. A node-disjoint multipath routing protocol based on AODV. In *Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on*, pages 312–316. IEEE, 2010.

[27] D. Djenouri, L. Khelladi, and N. Badache. A survey of security issues in mobile ad hoc networks. *IEEE communications surveys*, 7(4), 2005.

[28] Talipov E. A. 2009. NS2: Adding malicious node to AODV, [online] available at: http://elmurod.net/en/index.php/archives/196, retrieved:, May 2015.

[29] M. A. El-Gendy, A. Bose, and K. G. Shin. Evolution of the internet QoS and support for soft real-time applications. *Proceedings of the IEEE*, 91(7):1086–1104, 2003.

[30] S. Gangwar, S. Pal, and K. Kumar. Mobile ad hoc networks: A comparative study of QoS routing protocols. *Arxiv preprint arXiv: 1201.5688*, 2012.

[31] M. Gerharz, C. Vogt, and C. De Waal. Current approaches towards improved quality-of-service provision in mobile ad-hoc networks. *Computer Science Department, Communications Systems*, 2003.

[32] P. Ghosekar, G. Katkar, and P. Ghorpade. Mobile ad hoc networking: imperatives and challenges. *International Journal of Computer Applications IJCA*, (3):153–158, 2010.

[33] K. Govindan and P. Mohapatra. Trust computations and trust dynamics in mobile adhoc networks: a survey. *Communications Surveys & Tutorials, IEEE*, 14(2):279–298, 2012.

[34] Qijun Gu. Packet-dropping attack. In HenkC. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 899–902. Springer US, 2011.

[35] L. Hanzo. *Quality of Service-Aware Routing and Admission Control for Mobile Ad Hoc Networks*. PhD thesis, University of Surrey, 2009.

[36] L. Hanzo and R. Tafazolli. A survey of QoS routing solutions for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 9(2 2nd Quarter):50–70, 2007.

[37] Wenbo He and K. Nahrstedt. An integrated solution to delay and security support in wireless networks. In *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, volume 4, pages 2211–2215, April 2006.

[38] J. Hoebeke, I. Moerman, B. Dhoedt, and P. Demeester. An overview of mobile ad hoc networks: applications and challenges. *JOURNAL-COMMUNICATIONS NETWORK*, 3(3):60–66, 2004.

[39] J. Hu and M. Burmester. LARS: a locally aware reputation system for mobile ad hoc networks. In *Proceedings of the 44th annual Southeast regional conference*, pages 119–123. ACM, 2006.

[40] Yih-Chun Hu, Adrian Perrig, and David B Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2nd ACM workshop on Wireless security*, pages 30–40. ACM, 2003.

[41] Min Huang, Qinpei Liang, and Jianqing Xi. A parallel disjointed multi-path routing algorithm based on OLSR and energy in ad hoc networks. *Journal of Networks*, 7(4):613–620, 2012.

[42] Youngki Hwang and P. Varshney. An adaptive QoS routing protocol with dispersity for ad-hoc networks. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, page 10 pp., jan. 2003.

[43] P. Jacquet. Optimized link state routing protocol (OLSR). 2003.

[44] C. Jensen and P. Connell. Trust-based route selection in dynamic source routing. *Trust Management*, pages 150–163, 2006.

[45] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 195–206. ACM, 1999.

[46] D. Johnson, Y. Hu, and D. Maltz. The dynamic source routing protocol (DSR) for mobile ad hoc networks for ipv4. *RFC4728*, pages 2–100, 2007.

[47] SeokHo Jung, Elmurod Talipov, MyunWhan Ahn, and Chonggun Kim. A discovery method for node-disjoint multi-paths by valid source-destination edges. *Proceeding in Communication Systems and Networks*, 2007.

[48] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 243–254, New York, NY, USA, 2000. ACM.

[49] K. K. Kasera and R. Ramanathan. A location management protocol for hierarchically organized multihop mobile wireless networks. In *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, pages 158–162 vol.1, Oct 1997.

[50] Qifa Ke, David A Maltz, and David B Johnson. Emulation of multi-hop wireless ad hoc networks. In *Proceedings of the 7th International Workshop on Mobile Multimedia Communications (MoMuC 2000)*, 2000.

[51] Junhyung Kim, Jangkyu Yun, Mahnsuk Yoon, Keuchul Cho, Honggil Lee, and Kijun Han. A routing metric based on available bandwidth in wireless mesh networks. In *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, volume 1, pages 844–849, Feb 2010.

[52] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wirel. Netw.*, 6(4):307–321, July 2000.

[53] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET simulation studies: the incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(4):50–61, 2005.

[54] Chhagan Lal, Vijay Laxmi, and Manoj Singh Gaur. A node-disjoint multipath routing method based on AODV protocol for MANETs. In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 399–405. IEEE, 2012.

[55] LAR and DREAM implementation for ns. LAR. http://controls.ae.gatech.edu/claus/ns_lardream.html. retrieved:, 2014.

[56] S. B. Lee and A. T. Campbell. Insignia: In-band signaling support for QoS in mobile ad hoc networks. In *Proc of 5th International Workshop on Mobile Multimedia Communications (MoMuC, 98), Berlin, Germany.* Citeseer, 1998.

[57] S. J. Lee and M. Gerla. AODV-BR: Backup routing in ad hoc networks. In *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE*, volume 3, pages 1311–1316. Ieee, 2000.

[58] S. J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 10, pages 3201–3205. IEEE, 2001.

[59] Seoung-Bum Lee, Gahng-Seop Ahn, Xiaowei Zhang, and Andrew T Campbell. Insignia: An ip-based quality of service framework for mobile ad hoc networks. *Journal of Parallel and distributed Computing*, 60(4):374–406, 2000.

[60] R. Leung, Jilei Liu, E. Poon, A.-L.C. Chan, and Baochun Li. MP-DSR: a QoS-aware multi-path dynamic source routing protocol for wireless ad-hoc networks. In *Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on*, pages 132–141, 2001.

[61] Meng Li, Lin Zhang, Victor OK Li, Xiuming Shan, and Yong Ren. An energy-aware multipath routing protocol for mobile ad hoc networks. *ACM Sigcomm Asia*, 5:10–12, 2005.

[62] Zhi Li and Yu-Kwong Kwok. A new multipath routing approach to enhancing tcp security in ad hoc wireless networks. In *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*, pages 372–379. IEEE, 2005.

[63] W. H. Liao, S. L. Wang, J. P. Sheu, and Y. C. Tseng. A multi-path QoS routing protocol in a wireless mobile ad hoc network. *NetworkingICN 2001*, pages 158–167, 2001.

[64] C. R. Lin. On-demand QoS routing in multihop mobile networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1735–1744. IEEE, 2001.

[65] X. Lin and I. Stojmenovic. Location-based localized alternate, disjoint and multi-path routing algorithms for wireless networks. *Journal of Parallel and Distributed Computing*, 63(1):22–32, 2003.

[66] K. Liu, N. Abu-Ghazaleh, and K. D. Kang. Location verification and trust management for resilient geographic routing. *Journal of Parallel and Distributed Computing*, 67(2):215–228, 2007.

[67] Kejun Liu, Jing Deng, P. K. Varshney, and K. Balakrishnan. An acknowledgment-based approach for the detection of routing misbehavior in MANETs. *Mobile Computing, IEEE Transactions on*, 6(5):536–550, May 2007.

[68] Valeria Loscri and Salvatore Marano. A new geographic multipath protocol for ad hoc networks to reduce the route coupling phenomenon. In *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd*, volume 3, pages 1102–1106. IEEE, 2006.

[69] W. Lou, W. Liu, Y. Zhang, and Y. Fang. Spread: Improving network security by multipath routing in mobile ad hoc networks. *Wireless Networks*, 15(3):279–294, 2009.

[70] B. Lu. *Quality of Service (QoS) security in mobile ad hoc networks.* PhD thesis, Texas A&M University, 2005.

[71] K. Majumder, S. Ray, and S. K. Sarkar. Implementation and performance analysis of the gateway discovery approaches in the integrated manet-internet scenario. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pages 601–605. IEEE, 2011.

[72] M. K. Marina and S. R. Das. On-demand multipath distance vector routing in ad hoc networks. In *Network Protocols, 2001. Ninth International Conference on*, pages 14–23, Nov 2001.

[73] M. K. Marina and S. R. Das. Ad hoc on-demand multipath distance vector routing. *Wireless Communications and Mobile Computing*, 6(7):969–988, 2006.

[74] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network, IEEE*, 15(6):30–39, 2001.

[75] Peter J. J. McNerney and Ning Zhang. A 2-dimensional approach to QoS provisioning in adversarial mobile ad hoc network environments. In *Proceedings of the 15th ACM International Conference on Modeling, Analysis and*

Simulation of Wireless and Mobile Systems, MSWiM '12, pages 143–150, New York, NY, USA, 2012. ACM.

[76] E. Mohanraj and K. Duraiswamy. Integrated security and QOS routing on data communication framework in mobile ad hoc networks. *European Journal of Scientific Research*, 72(3):383–393, 2012.

[77] P. Mohapatra, J. Li, and C. Gui. QoS in mobile ad hoc networks. *IEEE Wireless Communications*, 10(3):44–53, 2003.

[78] S. Mueller, R. Tsang, and D. Ghosal. Multipath routing in mobile ad hoc networks: Issues and challenges. *Performance Tools and Applications to Networked Systems*, pages 209–234, 2004.

[79] A. Munaretto, H. Badis, K. Al Agha, and G. Pujolle. A link-state QoS routing protocol for ad hoc networks. In *Mobile and Wireless Communications Network, 2002. 4th International Workshop on*, pages 222 – 226, 2002.

[80] A. Nasipuri, R. Castañeda, and S. R. Das. Performance of multipath routing for on-demand protocols in mobile ad hoc networks. *Mobile Networks and Applications*, 6(4):339–349, 2001.

[81] T. A. Nguyen, T. A. Yang, A. Perez-Davila, and M. W. Ding. *Evaluations of secure MANET routing protocols in malicious environments.* PhD thesis, University of Houston-Clear Lake, 2006.

[82] E. Nordstrom, P. Gunningberg, and H. Lundgren. A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on*, pages 100–109, Feb 2005.

[83] R. Ogier, F. Templin, and M. Lewis. Topology dissemination based on reverse-path forwarding (TBRPF). Technical report, RFC Editor, 2004.

[84] Kitisak Osathanunkul. A cross-layer and multi-metric routing decision making framework for MANETs. 2013.

[85] Kitisak Osathanunkul and Ning Zhang. A countermeasure to black hole attacks in mobile ad hoc networks. In *International Conference on Networking, Sensing and Control*, 2011.

[86] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure message transmission in mobile ad hoc networks. *Ad Hoc Networks*, 1(1):193 – 209, 2003.

[87] L Felipe Perrone. Modeling and simulation best practices for wireless ad hoc networks. In *Simulation Conference, 2003. Proceedings of the 2003 Winter*, volume 1, pages 685–693. IEEE, 2003.

[88] A. A. Pirzada and C. McDonald. Trust establishment in pure ad-hoc networks. *Wireless Personal Communications*, 37(1):139–168, 2006.

[89] R. Plestys and R. Zakarevicius. A testbed for performance evaluation of mobile ad hoc network. In *Information Technology Interfaces (ITI), 2010 32nd International Conference on*, pages 155–160, June 2010.

[90] BS Pradeep. A new method for load balancing and QOS in on demand protocols–in the MANETs perspective.

[91] R. Prasad, C. Dovrolis, M. Murray, and K. Claffy. Bandwidth estimation: metrics, measurement techniques, and tools. *Network, IEEE*, 17(6):27–35, Nov 2003.

[92] L. K. Qabajeh, M. L. M. Kiah, and M. Qabajeh. A qualitative comparison of position-based routing protocols for ad-hoc networks. *International Journal of Computer Science and Network*, 9(2):131–140, 2009.

[93] A. K. Rai, R. R. Tewari, and S. K. Upadhyay. Different types of attacks on integrated manet-internet communication. *International Journal of Computer Science and Security (IJCSS) Volume (4)*, (3):265–274, 2010.

[94] I. Rai. *QOS support in edge routers.* PhD thesis, PhD thesis, Télécom Paris, Institut Eurecom, France, 2004.

[95] T. B. Reddy, I. Karthigeyan, BS Manoj, and C. Murthy. Quality of service provisioning in ad hoc wireless networks: a survey of issues and solutions. *Ad Hoc Networks*, 4(1):83–124, 2006.

[96] Frank E. Ritter, Michael J. Schoelles, Karen S. Quigley, and Laura Cousino Klein. *Human-in-the-Loop Simulations: Methods and Practice*, chapter Determining the Number of Simulation Runs: Treating Simulations as Theories by Not Sampling Their Behavior, pages 97–116. Springer London, London, 2011.

[97] E. M. Royer and C. E. Perkins. An implementation study of the AODV routing protocol. In *Wireless Communications and Networking Confernce, 2000. WCNC. 2000 IEEE*, volume 3, pages 1003–1008. IEEE, 2000.

[98] E. M. Royer and C. K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *Personal Communications, IEEE*, 6(2):46–55, 1999.

[99] G. Santhi. A survey of QoS routing protocols for mobile ad hoc networks. 2010.

[100] P. Sethi and G. Barua. CRESQ: providing QoS and security in ad hoc networks. In *Parallel, Distributed and Network-Based Processing, 2003. Proceedings. Eleventh Euromicro Conference on*, pages 544–550. IEEE, 2003.

[101] Kunal Shah. *Performance analysis of mobile agents in wireless internet applications using simulation.* PhD thesis, Lamar University, 2003.

[102] S. H. Shah and K. Nahrstedt. Predictive location-based QoS routing in mobile ad hoc networks. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 2, pages 1022–1027. IEEE, 2002.

[103] Priyanka Sharma, Kamal Sharma, and Surjeet Dalal. Reviewing MANET network security threats. *identity*, pages 25–30, 2014.

[104] ZhengMing Shen and J. P. Thomas. Security and QoS self-optimization in mobile ad hoc networks. *Mobile Computing, IEEE Transactions on*, 7(9):1138–1151, Sept 2008.

[105] Tao Shu and M. Krunz. Privacy-preserving and truthful detection of packet dropping attacks in wireless ad hoc networks. *Mobile Computing, IEEE Transactions on*, 14(4):813–828, April 2015.

[106] The Network Simulator. NS-2. http://nsnam.isi.edu/nsnam/index. retrieved:, May 2014.

[107] R. Sivakami and G. M. K. Nawaz. Reliable communication for MANETs in military through identification and removal of byzantine faults. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 5, pages 377–381, April 2011.

[108] P. Stüdi. *Quality of service for mobile ad hoc networks.* PhD thesis, 2003.

[109] PK Suri, MK Soni, and P. Tomar. Cluster based QoS routing protocol for MANET.

[110] T. Taleb, Y. H. Aoul, and A. Benslimane. Integrating security with QoS in next generation networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5, Dec 2010.

[111] S. Taneja and A. Kush. A survey of routing protocols in mobile ad hoc networks. *International Journal of innovation, Management and technology*, 1(3):2010–0248, 2010.

[112] A Chirag Tehlanl and Divya Sharma. A study on different security threats in mobile ad-hoc network. *International Journal of Information and Computation Technology. ISSN*, pages 0974–2239.

[113] M. Tekaya, N. Tabbane, and S. Tabbane. Multipath routing with load balancing and QoS in ad hoc network. *IJCSNS International Journal of Computer Science and Network Security*, 10(8):280–286, 2010.

[114] A. Tsirigos and Z. J. Haas. Multipath routing in the presence of frequent topological changes. *Communications Magazine, IEEE*, 39(11):132–138, 2001.

[115] Aishwarya Sagar Anand Ukey and Meenu Chawla. Detection of packet dropping attack using improved acknowledgement based scheme in MANET. *IJCSI International Journal of Computer Science Issues*, 7(4):12–17, 2010.

[116] I. Ullah and S. U. Rehman. Analysis of black hole attack on MANETs using different manet routing protocols. *Program Electrical Engineering with*

emphasis on Telecommunication, Type of thesis-Master Thesis, Electrical Engineering, Thesis no: MEE-2010-2698, 2010.

[117] N. S. M. Usop, A. Abdullah, and A. F. A. Abidin. Performance evaluation of AODV, DSDV & DSR routing protocol in grid environment. *IJCSNS International Journal of Computer Science and Network Security*, 9(7):261–268, 2009.

[118] M. Veerayya, V. Sharma, and A. Karandikar. SQ-AODV: A novel energy-aware stability-based routing protocol for enhanced QoS in wireless ad-hoc networks. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7. IEEE, 2008.

[119] L. Wang, Y. Shu, M. Dong, L. Zhang, and O. W. W. Yang. Adaptive multipath source routing in ad hoc networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 3, pages 867–871. IEEE, 2001.

[120] Lei Wang, Seungho Jang, and Tae young Lee. Redundant source routing for real-time services in ad hoc networks. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 7 pp.–87, Nov 2005.

[121] M. Wang. *MANET global connectivity and mobility management using HMIPV6 and OLSR*. PhD thesis, Carleton University, 2003.

[122] Kui Wu, Janelle Harms, et al. QoS support in mobile ad hoc networks. *Crossing Boundaries-the GSA Journal of University of Alberta*, 1(1):92–106, 2001.

[123] Hannan Xiao, W. K. G. Seah, A. Lo, and K. C. Chua. A flexible quality of service model for mobile ad-hoc networks. In *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 1, pages 445 –449 vol.1, 2000.

[124] K. Xu and M. Gerla. A heterogeneous routing protocol based on a new stable clustering scheme. In *MILCOM 2002. Proceedings*, volume 2, pages 838–843. IEEE, 2002.

[125] Yuan Xue and Klara Nahrstedt. Providing fault-tolerant ad hoc routing service in adversarial environments. *Wireless Personal Communications*, 29(3-4):367–388, 2004.

[126] Zhenqiang Ye, Srikanth V. Krishnamurthy, and Satish K. Tripathi. A framework for reliable routing in mobile ad hoc networks. In *IEEE INFOCOM*, pages 270–280, 2003.

[127] Zhuxiu Yuan, Lei Wang, Cheng Meng, Chunlei Liu, T. Q. Duong, and Lei Shu. Analysis on capacity and delay for redundant multiple source routing in mobile ad hoc networks. In *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pages 158–163, Dec 2010.

[128] Theodore Zahariadis, Panagiotis Trakadas, HelenC. Leligou, Sotiris Maniatis, and Panagiotis Karkazis. A novel trust-aware geographical routing scheme for wireless sensor networks. *Wireless Personal Communications*, 69(2):805–826, 2013.

[129] M. Zeshan, S. A. Khan, A. R. Cheema, and A. Ahmed. Adding security against packet dropping attack in mobile ad hoc networks. In *Future Information Technology and Management Engineering, 2008. FITME '08. International Seminar on*, pages 568–572, Nov 2008.

[130] Yujun Zhang, Tan Yan, Jie Tian, Qi Hu, Guiling Wang, and Zhongcheng Li. Tohip: A topology-hiding multipath routing protocol in mobile ad hoc networks. *Ad Hoc Networks*, 21(0):109 – 122, 2014.

[131] W. Zhao, D. Olshefski, and H. Schulzrinne. Internet quality of service: An overview. *Columbia University, New York, New York, Technical Report CUCS-003-00*, 2000.